

\$19.95

HOW TO

WORKBOOKS FROM IDBMA, INC.

ACCESS

by Harvey Eric Rodstein



*Terri Hale
1213 Blue Gum Lane
Newport Beach, CA 92660*

**“How To”
Workbooks From
IDBMA, Inc.**

ACCESS

By Harvey Eric Rodstein

© Copyright IDBMA, Inc. 1989
All rights reserved
Printed in the United States

PICK and PICK Operating System are registered trademarks of Pick Systems of Irvine, California.

Computer layout and design: S. Patrice Makovic
Copy Editor: Maureen McCarthy
Cover design: Cheri DeBusk, Chay Parra

Copyright 1989 by IDBMA, Inc. All rights reserved, including the right of reproduction in whole, or in part, in any form without the prior written consent of the copyright owner. Published by International Database Management Association, Inc., 10675 Treena Street, Suite 103, San Diego, California 92131. International Spectrum is a registered trademark of IDBMA, Inc. PICK is a registered trademark of Pick Systems, Inc. of Irvine, California. "How To" workbooks are trademarks of IDBMA, Inc. of San Diego, California.

Printed in the United States of America

ISBN 0-942093-05-4

To my wife and my best friend,
all wrapped up in one single
package... haneczka.

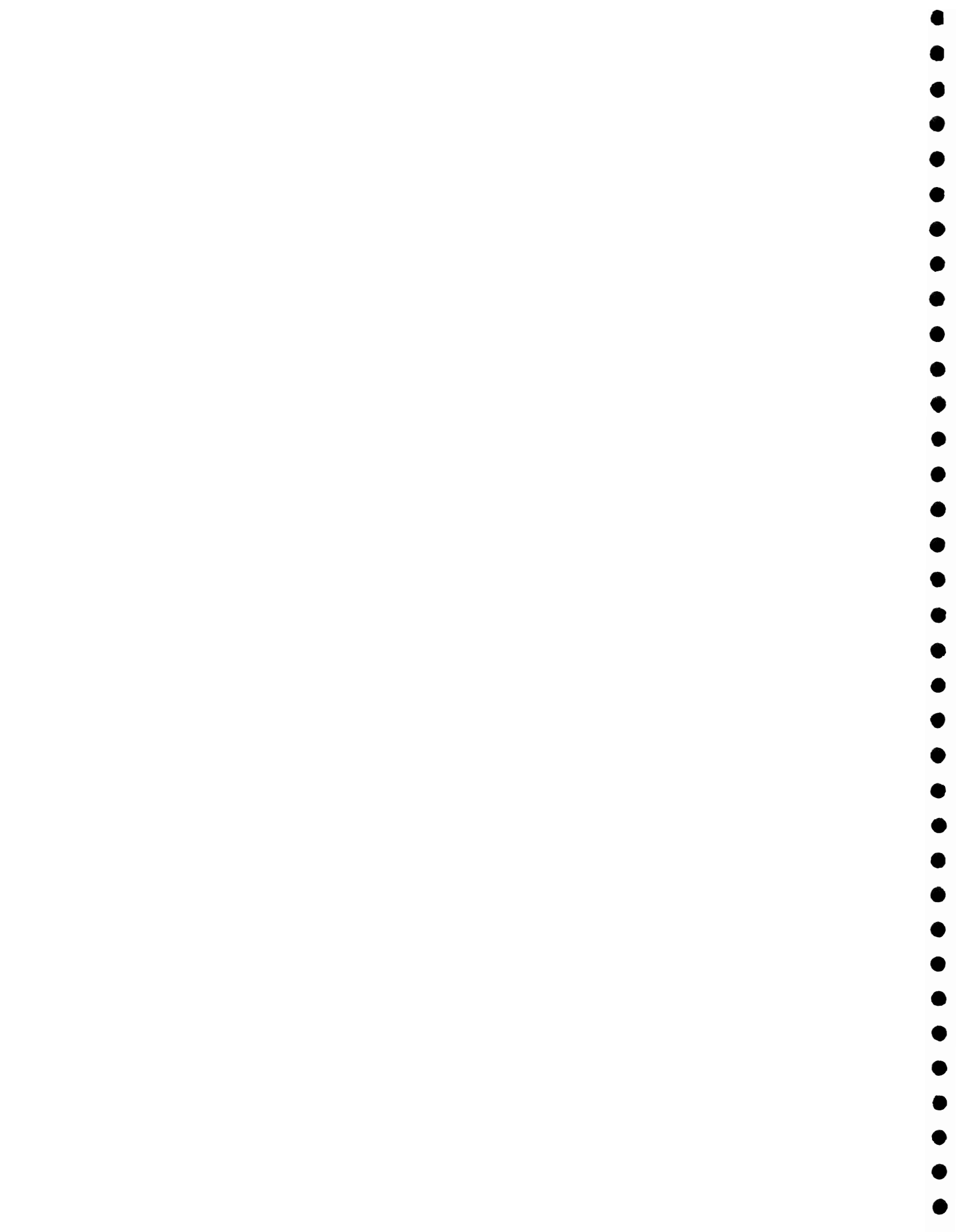
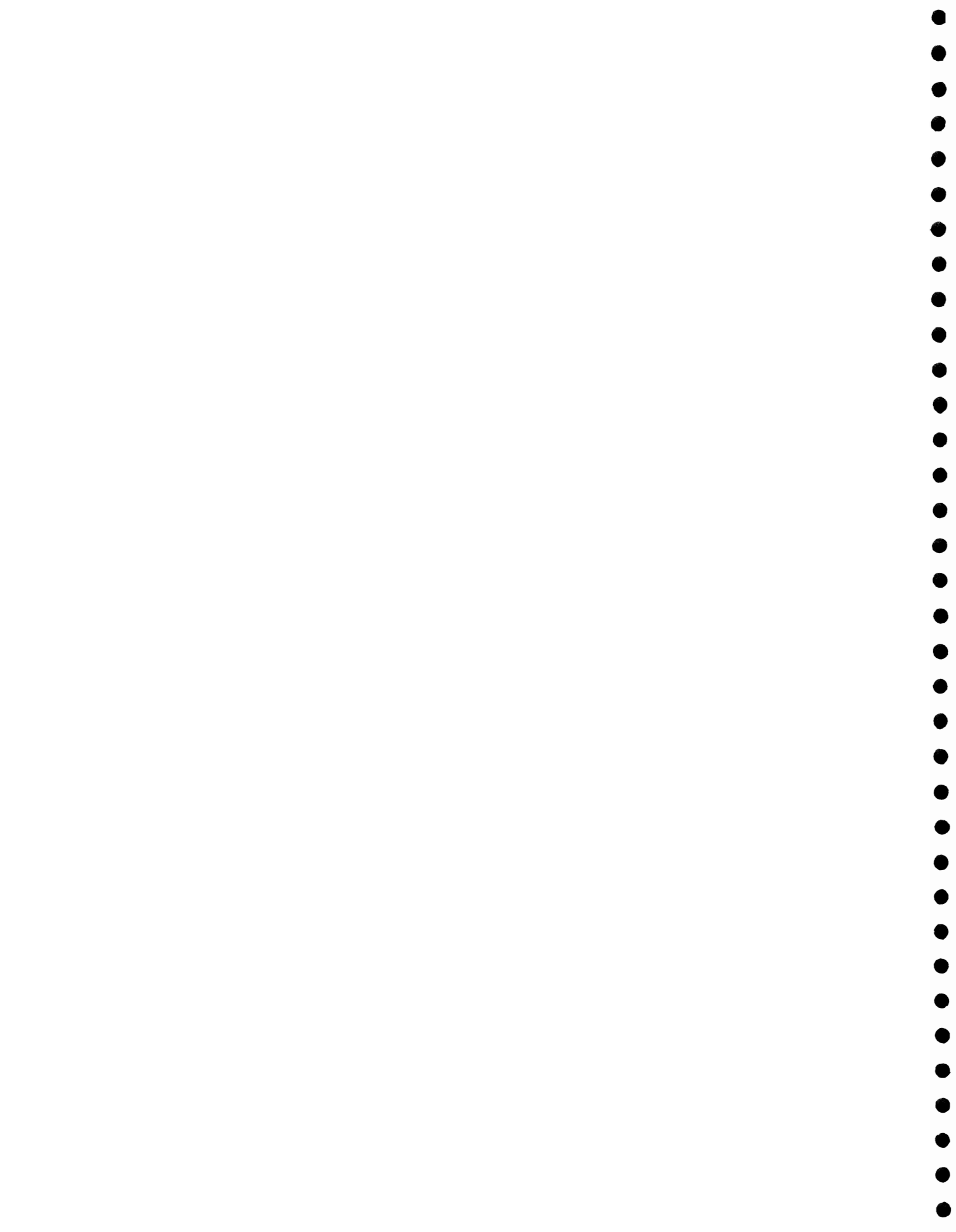


TABLE OF CONTENTS

ABOUT ACCESS	1
1. SYSTEM OVERVIEW	3
1.1 THE SYSTEM	
1.2 ACCOUNTS	
1.3 FILES	
1.4 ITEMS	
1.4.1 <i>Item-ids</i>	
1.4.2 <i>Attributes.</i>	
1.5 DATA DEFINITION ITEMS	
2. GETTING STARTED.	9
2.1 LOGGING ON	
2.2 LOOKING AROUND	
2.2.1 <i>Looking at Data Files</i>	
2.2.2 <i>Looking at Dictionaries</i>	
2.3 SETTING UP WORK FILES	
2.3.1 <i>Creating Work Files</i>	
2.3.2 <i>Entering Sample Data</i>	
2.3.3 <i>Entering Simple Data Definitions</i>	
3. THE ACCESS COMMAND SENTENCE.	23
3.1 THE VERB	
3.2 THE FILENAME	
3.3 ITEM-ID LIST	
3.4 OUTPUT DATA FIELDS	
3.5 THE SEQUENCE CLAUSE	
3.6 SELECTION CLAUSE	
3.6.1 <i>Selection Clause Structure</i>	
3.6.2 <i>Multiple Selection Clauses</i>	
3.6.3 <i>Value String Searching</i>	
3.6.4 <i>Wildcard Searches</i>	
3.6.5 <i>ID Selection Criteria</i>	
3.7 OUTPUT MODIFIERS AND OPTIONS	
3.8 THROWAWAY CONNECTIVES	
4. COLUMNAR REPORTING.	45
4.1 THE BASICS	
4.2 EXPLICIT OUTPUT FORMATTING	
4.3 IMPLICIT OUTPUT FORMATTING	
4.4 HANDLING MULTIVALUED OUTPUT FORMATTING	
4.4.1 <i>Simple Output</i>	
4.4.2 <i>Sorted Output</i>	
4.5 CUSTOM HEADINGS AND FOOTINGS	

4.6	GENERATING REPORT BREAKS	
4.6.1	<i>Simple Breaks</i>	
4.6.2	<i>Totals and Sub-Totals</i>	
4.6.3	<i>Grand Totals</i>	
4.6.4	<i>BREAK Options</i>	
4.7	OTHER OUTPUT MODIFIERS	
4.8	CHOOSING THE OUTPUT DEVICE	
5.	MAILING LABELS	73
5.1	LABEL VERBS AND PARAMETERS	
5.2	USING OUTPUT MODIFIERS	
5.3	USING INDENT	
6.	ACCESS LISTS	81
6.1	SELECTING AND USING LISTS	
6.2	CATALOGUED LISTS	
6.2.1	<i>Saving Lists</i>	
6.2.2	<i>Retrieving Lists</i>	
6.3	REPORTING CATALOGUED LISTS	
6.4	LIST MAINTENANCE	
6.4.1	<i>Copying Lists</i>	
6.4.2	<i>Deleting Lists</i>	
6.4.3	<i>Editing Lists</i>	
7.	FORMATTING WITH DATA DEFINITIONS	99
7.1	INTRODUCTION	
7.2	COLUMN HEADERS	
7.3	JUSTIFICATION	
7.4	LENGTH	
7.5	HIDDEN COLUMNS	
8.	CONVERSIONS AND CORRELATIVES	113
8.1	OVERVIEW	
8.2	DATE AND TIME CONVERSIONS	
8.2.1	<i>The D Conversion</i>	
8.2.2	<i>The MT Conversion</i>	
8.3	CHARACTER MASKING	
8.3.1	<i>The MC Conversion</i>	
8.4	STRING MASKING	
8.5	MONEY (DECIMAL) MASKING	
8.6	STRING EXTRACT CORRELATIVES	
8.6.1	<i>The Text Extract (T)</i>	
8.6.2	<i>The Group Extract (G)</i>	
8.7	FILE TRANSLATE CORRELATIVE	
8.7.1	<i>The Translate Syntax</i>	
8.7.2	<i>A Hypothetical Case</i>	
8.8	MULTIPLE CONVERSIONS AND CORRELATIVES	
8.9	CONCATENATION CORRELATIVE	
8.10	FUNCTION CORRELATIVE	
8.10.1	<i>The F Correlative Syntax</i>	
8.10.2	<i>Arithmetic Scaling</i>	
8.10.3	<i>Concatenation</i>	
8.10.4	<i>The Simplest F Correlative</i>	
8.10.5	<i>Other Operands and Operators</i>	

8.11	ALGEBRAIC CORRELATIVE	
8.11.1	<i>Introduction to the A Correlative</i>	
8.11.2	<i>The Recursive Operand</i>	
8.11.3	<i>Other Special Operands</i>	
8.12	CALCULATIONS AT BREAK LINES	
9.	STORED ACCESS COMMAND SENTENCES	161
9.1	THE PROC ITEM	
9.2	STORING AND PROCESSING COMMAND SENTENCES	
9.2.1	<i>The Primary Output Buffer</i>	
9.2.2	<i>The Secondary Output Buffer</i>	
	EXERCISE ANSWERS	173



ABOUT ACCESS

ACCESS is a results oriented data retrieval language which is provided as an integral part of a PICK-based operating system. ACCESS is classified as a non-procedural language, meaning that you ask for what you want without telling the system the steps of how to get there. With ACCESS, you simply enter an English-like sentence describing the content, restrictions, and order that data is to be output. Output can be formatted as columnar reports or mailing labels and routed to a video display terminal, system printer, or tape backup device. ACCESS can also be used to build and save index lists to drive other ACCESS or TCL processes.

For the queezy who are new to the computer game, cheer up! Mistakes don't hurt a thing. ACCESS is for data retrieval, not update. Any slip-ups that you might make will not hurt the database.

COURSE OBJECTIVES

The purpose of this workbook is to help the PICK user master the basic elements of ACCESS. It is designed as a self-study tool, consisting of step-by-step explanations and examples. Fill-in-the-blank and multiple choice exercises punctuate the important topics.

Students are encouraged to follow along on their systems with an already existing applications database, or by entering the examples in this workbook. However, not all the examples are hands-on. Please note, that those examples in the rounded-corner boxes are intended to be interactive, and those in the square boxes are for observation only!



1

SYSTEM OVERVIEW

1.1 THE SYSTEM

The *system*, also known as the *system dictionary*, can be likened to a "records room" full of file cabinets with each containing information that is logically stored by account.

The file, SYSTEM, contains elements which define all the valid accounts. Before you can do any work, you must tell the system your account name. This process is called logging on. Logging on unlocks the file cabinet.

1.2 ACCOUNTS

An *account* is a directory of information and services. It can be likened to a file cabinet with a virtually unlimited number of drawers. These "drawers" (files) hold information grouped according to purpose (Inventory, Payables, Programs). Each department in a business may have its own account from which to work. For example, the billing department may have an account called BILLING, while the shipping department may work from the account called SHIPPING. The single most important account for system operations is called SYSPROG from which all system utilities, such as tape backup, port and printer configuration, and system verification are invoked.

Each account has a single master dictionary. The master dictionary contains elements that define all of the files and commands which are accessible in the account.

1.3 FILES

A *file* is the cabinet drawer. However, this drawer will keep pulling out forever, like a Three Stooges prop. The size of a file is limited only by the amount of disk available. A file is a set of distinct but related elements of information and is given a name to reflect its use. For example, orders may be kept in a file called ORDERS, while inventory data may be kept in a file called INVENTORY.

A file is actually made up of two files or "levels." The "upper" level is the DICTIONARY level. The "lower" level is the DATA level.

The DATA level contains the data and/or text.

The DICTIONARY level is exactly as it sounds, "a place to look up the meaning of words." Each field or group of fields in the data area can be described by a word in the dictionary. These "words" are called *data definition items*. These items are used by ACCESS for retrieving and formatting data.

Items are the file "folders." Each distinct element in a file is called an item.

Items contain multiple pieces of information. For example, each item in a file named CUSTOMERS represents a single customer. All of the information needed to process this customer is contained within the item. This may include name, address, city, state and zip. Each piece of information is a "field." In the PICK environment, fields are also called "attributes."

Many types of information can be stored in an item. Items can consist of fields of data (name, address, city, etc.), lines of descriptive text, program code statements (BASIC, PROC, or ASSEMBLER) or data retrieval information (DICTIONARY Data Descriptors).

All data within an item is stored as ASCII characters. (ASCII stands for American Standard Code for Information Interchange.) Please note that in most implementations, a maximum of 32,000 characters is allowed per item.

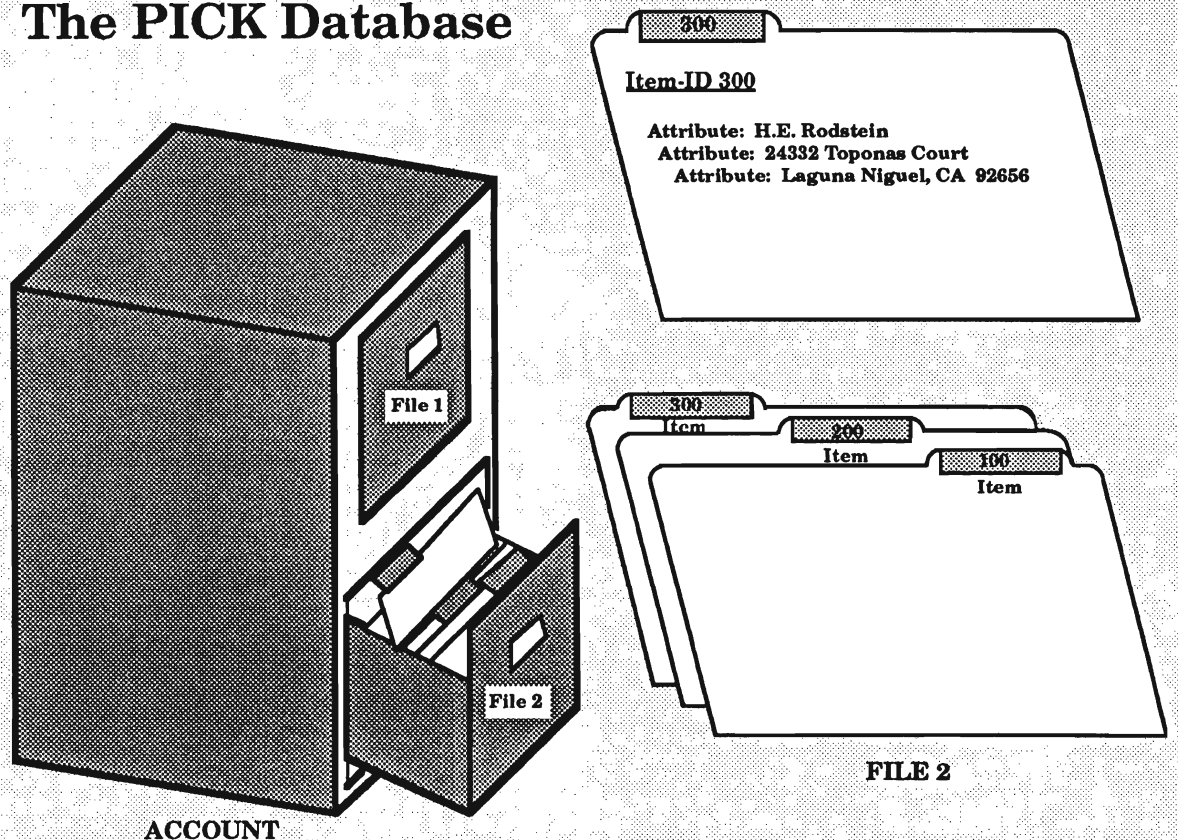
1.4.1

Item-Ids

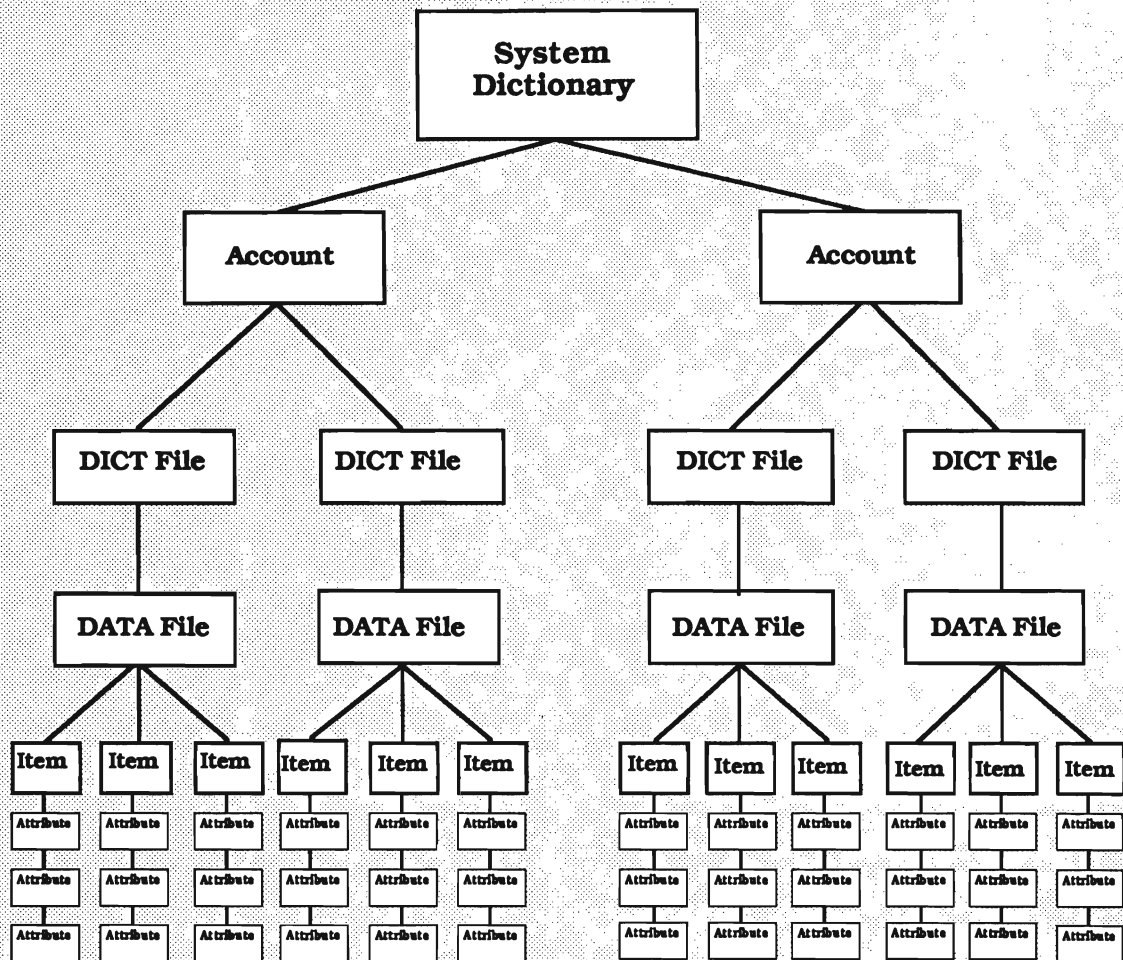
An item is assigned a free-form name. This name (IDentifier) is not unlike a reference tab on a file folder. *Item-ids* are used to uniquely identify items within a file and cannot be duplicated. The process used to address an item in a file is called "hashing." Hashing converts the item-id directly into location in the file. The item-id is actually used to map the physical disk address.

Be that as it may, all that you have to remember is the name of the item (item-id).

The PICK Database



Hierarchical Diagram of the System



1.4.2

Attributes

The fields of information which make up an item are called *attributes*.

Attributes vary in length.

A sample item in logical format:

```

ID: 100

001 Caesar, Julius
002 1 Appian Way
003 Olympia
004 WA
005 10101
  
```

There are five attributes in item 100 above.

The EDIT and COPY verbs present an item in logical format. Each attribute is displayed on a separate, labeled line. The actual physical layout of an item is somewhat different.

```
100^Caesar, Julius^ Appian Way^Olympia^WA^10101^__
```

Attributes are marked internally by a special reserved character. This character is called an *attribute mark*. (For those who care, an attribute mark is ASCII character 254.) ACCESS allows a name to be associated with each attribute.

Attributes can be further divided into fields called "values." An attribute containing many values is said to be a "multivalued" attribute. Values can be further divided into "sub-values."

Values are marked with value marks and sub-values with sub-value marks. Value and sub-value marks display on an EDIT line as a right bracket (]) and a back-slash (\), respectively. (A value mark is ASCII character 253 and a sub-value mark is ASCII character 252.)

```
003 101]201]505]303
```

```
:---:---:---:---:
```

Line 3 contains four values. The right brackets (]) represent value marks.

```
004 3\10\4]2\5\5
```

```
:-----:-----:
```

Line 4 contains two values with 3 sub-values each. The back-slashes represent sub-value marks.

The application programs already on your system take care of building these fields and keeping track of the attributes, values and sub-values. If, however, it is necessary to repair this data using the system editor, the delimiters must be generated from the terminal keyboard.

- Attribute marks are generated by a <ctl><shift><^>
- Value marks are generated by a <ctl><shift><]>
- Sub-value marks are generated by a <ctl><shift><\>

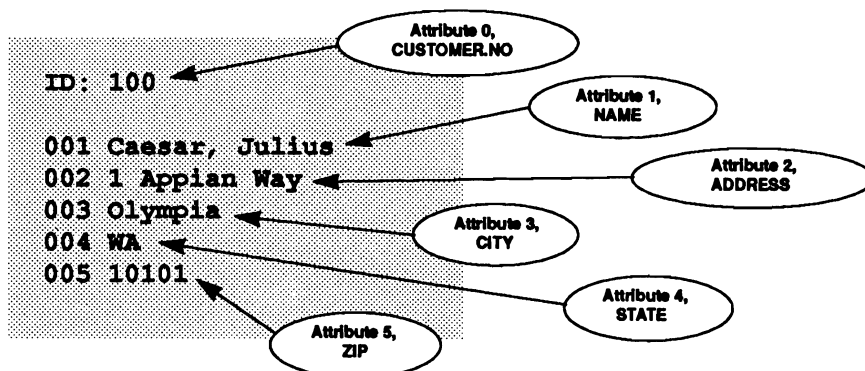
NOTE

Do not confuse the display of these special characters with the normal brackets and back-slashes. In fact, some terminals display these characters as totally different characters. For instance, the WYSE family of terminals display an attribute mark as a tilde (~), a value mark as a brace (}), and a sub-value mark as a broken line (!).

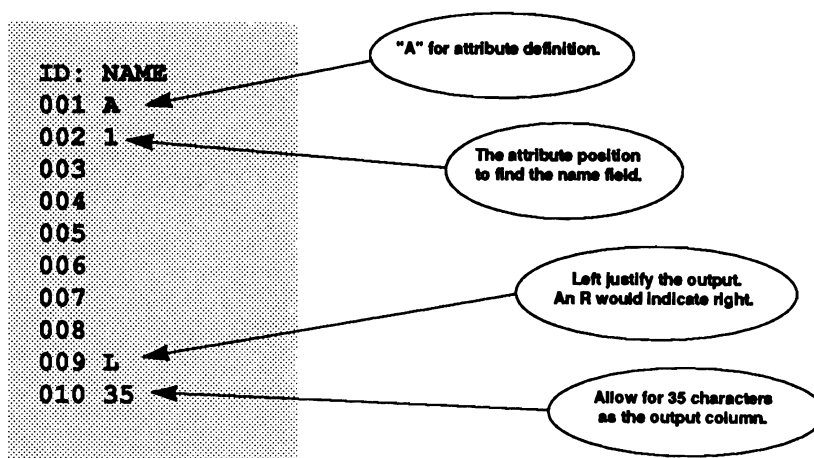
DATA DEFINITION ITEMS

A *data definition item* resides in the file dictionary and is assigned an item-id which is used as a descriptive word in ACCESS. The contents of the item are parameters which control the attribute position of the data field, output masking, justification, and length, and any calculations.

Definitions are used for data retrieval and do not affect data update.



The description of NAME is an item in the file dictionary which looks like this:



The other descriptive items are as follows:

```

ID: ADDRESS
001 A
002 2
003
004
005
006
007
008
009 L
010 45
  
```

```

ID: CITY
001 A
002 3
003
004
005
006
007
008
009 L
010 15
  
```

```

ID: STATE
001 A
002 4
003
004
005
006
007
008
009 L
010 2
  
```

```

ID: ZIP
001 A
002 5
003
004
005
006
007
008
009 R
010 10
  
```

EXERCISE

1

1. Files are grouped by _____.
2. _____ contains the definitions of every account.
3. Files and commands are defined in each account's _____.
4. Files are made up of _____.
5. Items are made up of lines of information called _____.
6. Items are retrieved by a unique key called the _____.
7. The _____ level holds the file's data.
8. The _____ level contains the data definition items.
9. Attributes are delimited by special characters called _____.
10. How many attributes can an item hold? _____
11. The maximum number of characters per item is _____.
12. Name two of the parameter functions that are stored in a data definition item.

13. True or False. Data definitions change what is actually stored on the file. _____
14. True or False. Data definition item-ids are words which describe data held in an attribute of an item. _____

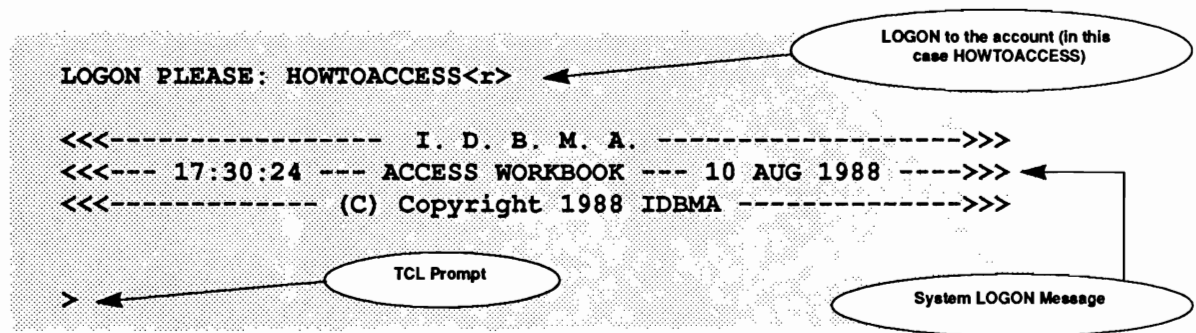
2

GETTING STARTED

2.1

LOGGING ON

Using an account name to LOGON to the system, gives you access to the group of files defined within that account.



You are now at TCL (Terminal Control Language).

The greater-than symbol (>) is the TCL prompt. It tells you that the system is ready for a command sentence to be entered.

NOTE

An account must exist before you can logon to the system. You may use an existing account or have one created for you. An account is created using the CREATE-ACCOUNT verb from the SYSPROG account. In the interest of getting to the point, the CREATE-ACCOUNT procedure is not covered in this workbook. Please consult your manuals for further details.

The first word of a command sentence is called a *verb*, since it invokes action. (The word "invoke" is used to mean "initiate.")

Some verbs may "stand alone." These are called simple TCL, or TCL-I verbs, since they invoke action which does not involve retrieving data from a file. Such verbs include: TIME, display the current time and date; WHO, display your port number and account name; or WHERE, list the ports logged onto the system.

Simple TCL sentence:

```
>verb<r>
```


Control Keys:

(Control keys are generated by depressing the <ctl> key simultaneously with any other key.)

<ctl>	Is the control key, used in conjunction with other keys to alter their function.
<ctl>H	Same as <backspace>.
<ctl>I	Same as <tab>.
<ctl>X	Restarts input at the beginning of the current line.
<ctl>W	Backs up input to the previous word.

Be careful with these:

<ctl>S	Is called X-OFF. It tells the system, "Don't send me anything until you hear from me again with an X-ON."
<ctl>Q	Is X-ON. It tells the system to start sending again!

If the <ctl>S is typed accidentally, the system appears to hang (stop processing). There is no response. Simply type a <ctl>Q (X-ON) before calling the DP department.

NOTE

Using "The Editor", typing any of the remaining control characters imbeds them within the text. There are no equivalent display characters for them. Therefore, EDIT displays a period (.) to mark the location of a <ctl> character. This can lead to problems since this "lint" is indistinguishable from an actual period.

There are a few other keys which generate characters different from the actual display. The following is a list of the characters which are displayed on most terminals. However, they are not standard. If your display differs, note the characters for future reference.

KEYS	DESCRIPTION
<escape>	Displays as a left-bracket ([).
<ctl><shift>6	Generates an attribute mark. Displays as an up-arrow (^).
<ctl><shift>]	The control, shifted right-bracket generates the multivalue delimiter, a value mark. This displays as a right-bracket (]).
<ctl><shift>\	The control, shifted back-slash (\) generates the sub-value delimiter, a sub-value mark. Displays as a back-slash (\).

2.2

LOOKING AROUND

Time to get your feet wet. The commands described in this section should help you navigate within an account.

2.2.1

Looking at Data Files

Once logged on to an account, it is necessary to get a listing of the files that are available. The verb to do this is LISTFILES.

At TCL type:

```
>LISTFILES<r>
```

The listing may look like this:

```
PAGE 1      *** MD *** FILE DEFINING ITEMS      20 AUG 1988

MD..... CODE F/BASE..... F/MOD..... F/SEP...

CUSTOMERS      D              6270              1      1
INVOICES       D              9456              3      1
STATES         D              8994              1      1
REPORTS        D              6147              1      1
ACC            Q              ACC              ACC
BLOCK-CONVERT  Q      BLOCK-CONVERT
ERRMSG         Q              SYSPROG          ERRMSG
M/DICT         Q
MD             Q
SYSTEM-ERRORS  Q      SYSTEM-ERRORS  SYSTEM-ERRORS

9 ITEMS LISTED.
```

Column header MD stands for Master Dictionary. All files are defined within an account Master Dictionary.

The column heading CODE tells you whether the file is defined locally on your account or is being used locally but defined on another account. The D represents a locally defined file. The Q represents a file defined on another account. The remaining columns provide information about the location and size of each file. Please refer to your system manuals for details.

It is now necessary to get a listing of what item-ids are already in the data-level of the file of your choice. The commands to do this are LIST and SORT.

```
>LIST filename<r>
```

or

```
>SORT filename<r>
```

LIST generates a listing of the item-ids in random order.

```
>LIST CUSTOMERS<r>

PAGE 1                      16:20:15   12 AUG 1988

CUSTOMERS.

106
107
100
101
102
103
104
105

8 ITEMS LISTED.
```

SORT generates a listing of the item-ids in alphabetical or numerical sequence.

```
>SORT CUSTOMERS<r>

PAGE 1                      08:13:39   20 FEB 1987

CUSTOMERS.

100
101
102
103
104
105
106
107

8 ITEMS LISTED.
```

The results of any of the above commands can be sent to the printer via the system spooler by using the option "P." Options must always be surrounded by parentheses.

```
>LISTFILES (P)<r>
>LIST filename (P)<r>
>SORT filename (P)<r>
```

2.2.2

Looking at Dictionaries

To be able to ask advanced questions about the data currently in a file, it is necessary to interrogate the DICTionary of the file to see what data descriptive words are available. A simple list of the data definition item-ids can be generated by the command sentences:

```
>LIST ONLY DICT filename<r>
```

```
>SORT ONLY DICT filename<r>
```

For the sample file CUSTOMERS, the command sentence and output might look like this:

```
>SORT ONLY DICT CUSTOMERS<r>

PAGE 1                                15:11:48    12 AUG 1988

CUSTOMERS..

ADDRESS
CITY
COMPANY
CONTACT
CUSTNUM
CUSTOMERS
STATE
ZIP

8 ITEMS LISTED.
```

A formatted report of the data definitions can be generated by using the "canned" system verb LISTDICT in the following format:

```
>LISTDICT filename<r>
```

or

```
>LISTDICT filename (P)<r>
```

Here is how LISTDICT looks when sent to the system printer.

CAUTION: The report may require a 132 output width defined for the printer.

```
>LISTDICT CUSTOMERS (P)<r>
```

Printed output:

PAGE 1			15:14:46 12 AUG 1988	
CUSTOMERS. CODE A/AMC S/NAME... S/AMC. CONVERSIONS....CORRELATIVES... TP MAX				
CUSTNUM	A	0	R	10
COMPANY	A	1	L	20
CONTACT	A	2	L	20
ADDRESS	A	3	L	20
CITY	A	4	L	15
STATE	A	5	L	2
ZIP	A	6	L	10
7 ITEMS LISTED.				

Each line in the report represents a data definition item in the file dictionary. Each important attribute in a definition item is assigned a column in the LISTDICT report.

COLUMN	DESCRIPTION
CODE	Attribute (A) types.
A/AMC	The attribute position pointer.
TP	The justification -- L for left, R for right.
MAX	The output length.

2.3

SETTING UP WORK FILES

If you wish to follow along with the examples in the workbook, it is necessary to set up the following work files. This set up is not required. You may use the files already available on your system.

2.3.1

Creating Work Files

The files needed are named CUSTOMERS, STATES, INVOICES and REPORTS.

Files are created on the current account by invoking the verb:

CREATE-FILE

The specifics on creating a new file are covered within the system manuals. For now, simply enter the following command sentences at TCL:

```
>CREATE-FILE CUSTOMERS 7,1 11,1<r>
```

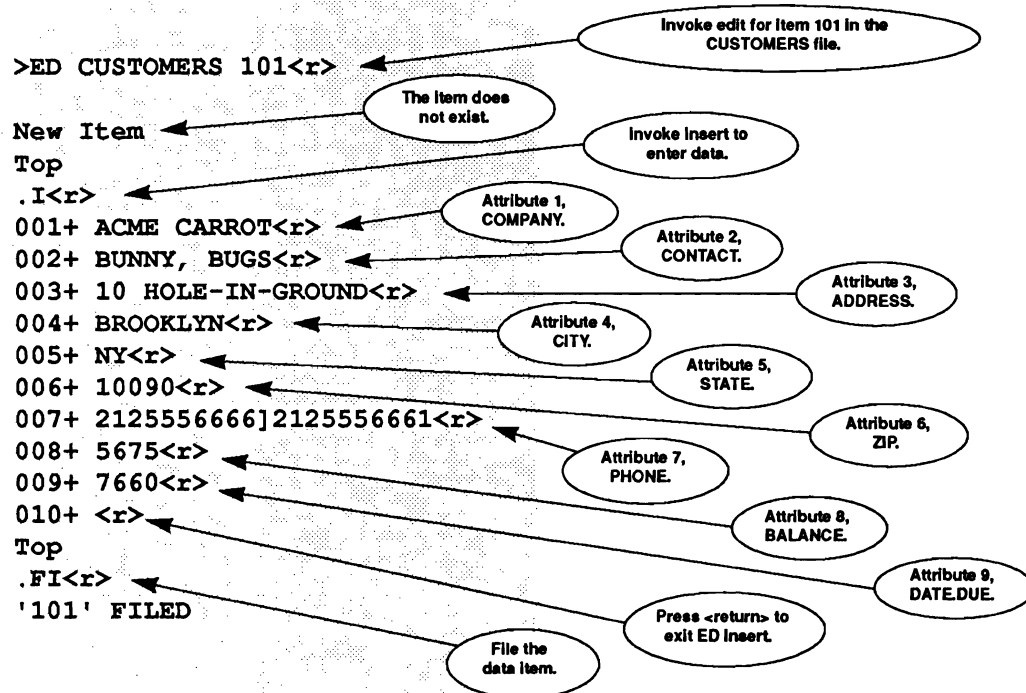
```
>CREATE-FILE STATES 7,1 11,1<r>
```

2.3.2

Entering Sample Data

The following sample items on page 9, plus any that you can think of, are used in the examples throughout the workbook. EDIT examples are given, but the mechanics are not explained. If you are not already familiar with EDIT, try the first workbook of the HOW TO series, "The Editor."

Enter the following sample item. (Remember: a rounded corner screen indicates a hands-on exercise and <r> indicates a carriage return.)



Please note that the right bracket (]) on attribute 007, PHONE numbers, is a value mark generated by a <shift><ctl><]>. In this case, Multivalues are used to store multiple phone numbers in the same attribute.

Notice that the PHONE is numeric only; the BALANCE contains no decimal points and the DATE.DUE follows no obviously logical structure for a date. These are internally formatted data fields. This is the storage format used by many applications. Don't worry about this just yet. The subject is covered in Chapter 8, "Conversions and Correlatives."

**SAMPLE
ITEMS**

Repeat the previous process for the following sample items:

106

001 PHINK, PHINK AND PHINK
002 PHINK, CLYDE
003 1600 PENNSYLVANIA AVE
004 SANTA ANA
005 CA
006 90234
007 2135551112
008 12675
009 8888

107

001 MONA LISA LTD
002 DAVINCI, LEONARDO
003 15 CENTURY DR
004 VENICE BEACH
005 CA
006 97771
007 2135554444]2135554321
008 78932
009 8543

100

001 SHIPS R US
002 HORNBLOWER, HORATIO
003 1110-1 W 57TH ST
004 NEW YORK
005 NY
006 10019
007 2125557676]2125556666
008 56780
009 7888

102

001 THE GUN STORE
002 FUDD, ELMER
003 30 WABBIT WAY
004 WEDONDO BEACH
005 CA
006 91234
007 2135559999
008 1234
009 -500

103

001 AEROBIC DEN
002 PORCELANA, PORCHE
003 139 HOLLYWOOD BLVD
004 HOLLYWOOD
005 CA
006 91329
007 2135557777
008 8912
009 7999

104

001 SPEAK EASY
002 PARTICIPLE, DANGLY
003 3 DAT DARE ST
004 GRAMMAR CITY
005 PA
006 12345
007 2155554433]2155556321]2155554444
008 2342
009 8555

105

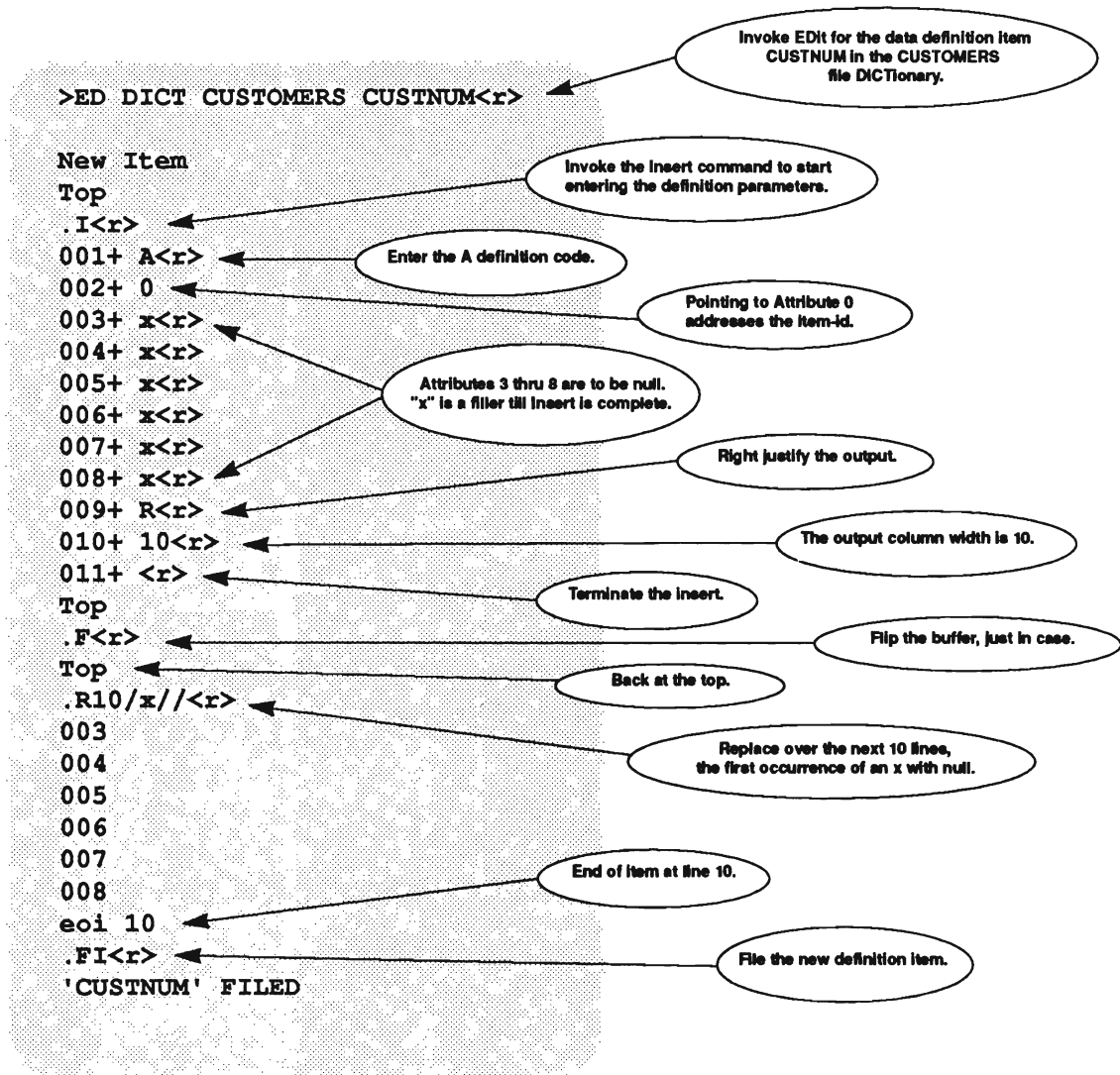
001 DIABOLICAL SOFTWARE
002 GROG, BILL SPIDER
003 01100 BINARY WAY
004 BYTELAND
005 CA
006 90001
007 4085554444
008 456
009 -1000

2.3.3

Entering Simple Data Definitions

To begin, enter the following items into the DICTIONary of the CUSTOMERS file. To be able to duplicate the samples in the workbook, these definition items must be entered. You may choose to skip this step and use your own existing database instead.

Follow this sample EDIT session.



Repeat the above procedure for the remaining data definition items on page 19.

SAMPLE ITEMS

COMPANY

001	A
002	1
003	
004	
005	
006	
007	
008	
009	L
010	20

CONTACT

001	A
002	2
003	
004	
005	
006	
007	
008	
009	L
010	20

ADDRESS

001	A
002	3
003	
004	
005	
006	
007	
008	
009	L
010	25

CITY

001	A
002	4
003	
004	
005	
006	
007	
008	
009	L
010	15

STATE

001	A
002	5
003	
004	
005	
006	
007	
008	
009	L
010	2

ZIP

001	A
002	6
003	
004	
005	
006	
007	
008	
009	L
010	10

PHONE

001	A
002	7
003	
004	
005	
006	
007	
008	
009	R
010	12

BALANCE

001	A
002	8
003	Balance Due
004	
005	
006	
007	MR2
008	
009	R
010	12

	DATE.DUE
001	A
002	9
003	
004	
005	
006	
007	D
008	
009	R
010	12

Both BALANCE and DATE.DUE have additional information in attribute 007. This is called an output conversion. It is placed there to translate what is called "internal data" to something that a human being can read. This is discussed further in Chapter 8, "Conversions and Correlatives." It's still too early in the game to talk about the care and feeding of definition items.

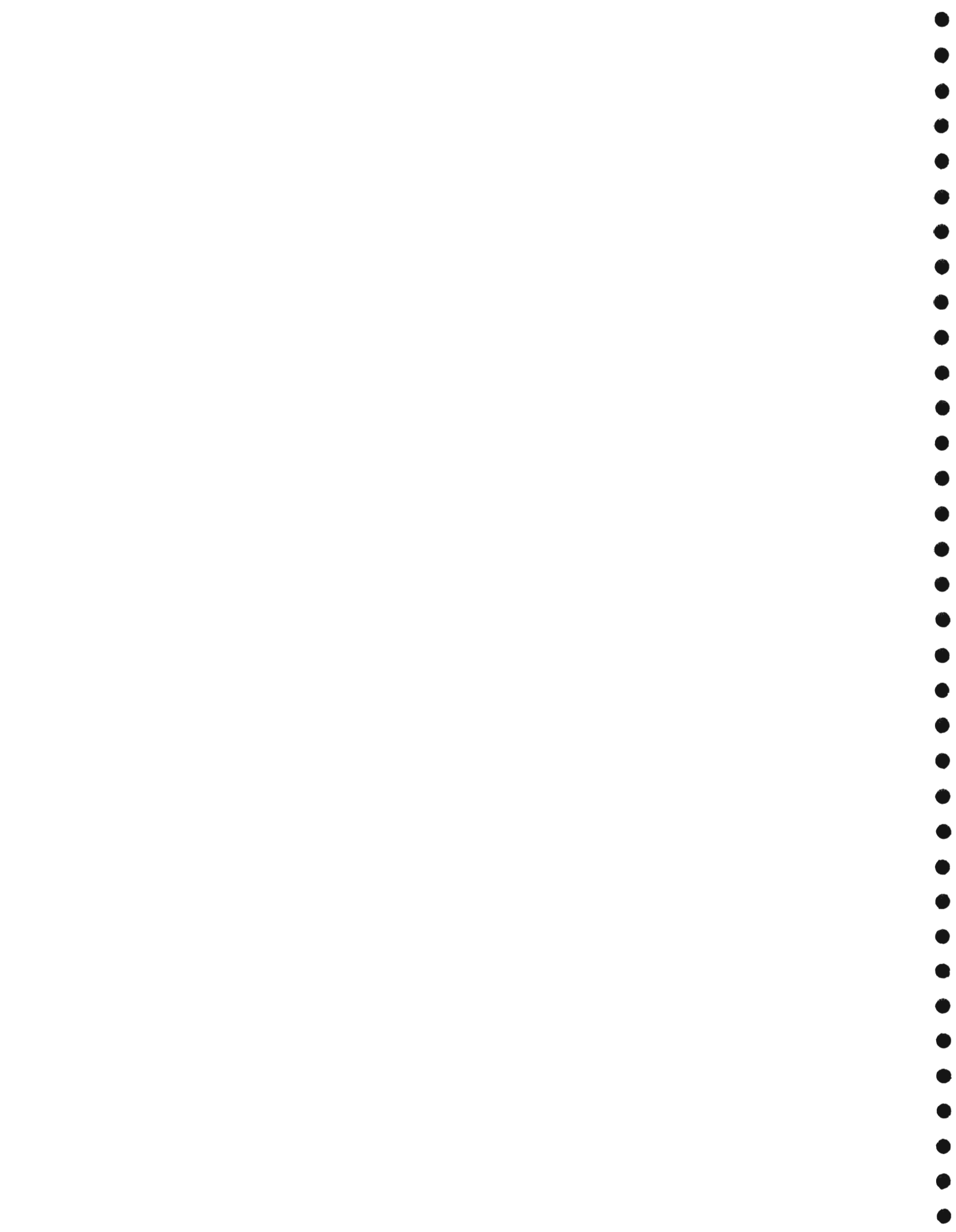
EXERCISE

2

1. A user gains access to a group of files on an account by _____.
2. TCL stands for _____.
3. Words which invoke action are called _____.
4. _____ produces a listing of all the files within an account.
5. _____ produces a random listing of item-ids in a file.
6. _____ produces a sorted listing of item-ids within a file.
7. Enter two command sentences to list the dictionary of the CUSTOMERS file.

 > _____

 > _____
8. What does the ">" represent? _____
9. In the LISTDICT output, the column header A/AMC means _____.



3

THE ACCESS COMMAND SENTENCE

This chapter could also be entitled, "The Grammar of ACCESS," since it deals with the structural elements which make up an ACCESS sentence.

The complete form of an ACCESS sentence:

```
>verb filename {item-id list}{sequence clause}{selection clause}{output-list}{modifiers}{(options)}
```

The basic components of an ACCESS statement are the verb and filename. As indicated by braces {}, the remaining elements are optional.

3.1

THE VERB

A verb is a word of action. The choice of verb controls the type of action; columnar reports, mailing labels, and index lists. Verbs travel in pairs. One generates an ordered output and the other generates a random output.

OUTPUT CATEGORY	VERB	DESCRIPTION
Columnar Reports	LIST	Random columnar listing.
	SORT	Ordered columnar listing.
Mailing Labels	LIST-LABEL	Random labels.
	SORT-LABEL	Ordered labels.
ACCESS Lists	SELECT	Random list selection.
	SSELECT	Sort and select. Ordered list selection.
Misc.	COUNT	The exception to the rule. Counts the number of items which pass a certain selection criteria. No output except for the final count.

These are the verbs used most often and concentrated on in this workbook. Please consult your manuals for a list of all of the verbs in ACCESS.

In a sentence, a verb needs an object. Since ACCESS is used to process items in files, the object must be a filename. An ACCESS verb must always be followed by a filename.

The filename specification is:

CUSTOMERS	The data level of CUSTOMERS.
DATA CUSTOMERS	Also, the data level of CUSTOMERS. The DATA modifier is not required.
DICTIONARY CUSTOMERS	The dictionary level of the CUSTOMERS file.

Try the examples presented in "Looking Around." (See Sec. 2.2.)

```
>LIST CUSTOMERS
or
>SORT CUSTOMERS
```

Possible TCL Errors:

If you should use another word which you think is a verb and it is not, the following error message is displayed:

```
[3] VERB?
```

If the filename you are interrogating does not exist, the following error message is displayed:

```
[10] FILENAME IS MISSING
```

In either case, check the spelling of the offending entry in the ACCESS statement. Notice that a number in brackets often appears before the actual message. This is the item-id of the error message in the ERRMSG file.

The simple form of the sentence has been demonstrated in Chapter 2, "Getting Started." However, in review, here is the sample output of a simple ACCESS sentence.

```
>SORT CUSTOMERS<r>
```

```
PAGE 1
```

```
08:01:15 16 AUG 1988
```

```
CUSTOMERS.
```

```
100
```

```
101
```

```
102
```

```
103
```

```
104
```

```
105
```

```
106
```

```
107
```

```
8 ITEMS LISTED.
```

Here is a sample of a sentence using the COUNT verb.

```
>COUNT CUSTOMERS<r>  
8 ITEMS COUNTED.
```

3.3

ITEM-ID LIST

Normally, ACCESS samples every item in a file. That's why large files take so long to generate output.

An ACCESS process can be restricted to a number of specific items if the item-ids are explicitly expressed in the sentence. The only requirement is that each item-id must be surrounded by a single quote (').

General form:

```
>verb filename 'item-id' 'item-id' ....
```

```
>LIST CUSTOMERS '101' '105'<r>  
PAGE 1                                09:33:01  13 AUG 1988  
CUSTOMERS.....  
101  
105  
2 ITEMS LISTED.
```

Possible Errors:

If you enter an item-id that is not on file, the following appears:

```
>LIST CUSTOMERS '100' '333'<r>  
PAGE 1                                09:33:01  13 AUG  
1988  
CUSTOMERS.....  
100  
'333' NOT ON FILE  
END OF LIST
```

← Notice that 333 does not
exist on file.

3.4

OUTPUT DATA FIELDS

The data to be output is listed by name within the ACCESS sentence.

```
>verb filename name1 name2 name3 ...
```

This is a random report showing the company name, city, and state.

```
>LIST CUSTOMERS COMPANY CITY STATE<r>
```

```
PAGE 1
```

```
10:40:30 13 AUG 1988
```

```
CUSTOMERS. COMPANY..... CITY..... STATE
```

106	PHINK, PHINK AND PHI NK	SANTA ANA	CA
107	MONA LISA LTD	VENICE BEACH	CA
100	SHIPS R US	NEW YORK	NY
101	ACME CARROT	BROOKLYN	NY
102	THE GUN STORE	WEDONDO BEACH	CA
103	AEROBIC DEN	HOLLYWOOD	CA
104	SPEAK EASY	GRAMMAR CITY	PA
105	DIABOLICAL SOFTWARE	BYTELAND	CA

```
8 ITEMS LISTED.
```

Sentence elements can be combined. Here, the item-list is combined with the output-list.

```
>LIST CUSTOMERS '101''105' COMPANY CITY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
------------------	-------------

LIST	verb
CUSTOMERS	filename
'101''105'	item-list
COMPANY CITY STATE	output-list

```
>LIST CUSTOMERS '101''105' COMPANY CITY STATE<r>
```

```
PAGE 1
```

```
10:40:30 13 AUG 1988
```

```
CUSTOMERS. COMPANY..... CITY..... STATE
```

101	ACME CARROT	BROOKLYN	NY
105	DIABOLICAL SOFTWARE	BYTELAND	CA

```
2 ITEMS LISTED.
```

Possible Errors:

If you enter a data definition item-id in the ACCESS statement, but the word is misspelled or does not exist, the following error message displays:

[24] THE WORD 'word' CANNOT BE IDENTIFIED

3.5 THE SEQUENCE CLAUSE

As within English grammar, ACCESS sentences may have modifying clauses. The sequence clause indicates the order in which data is to be output.

SORT class verbs order output by item-id unless a sequence clause is used. There is no effect when a sequence clause is used with a LIST class verb.

A sequence clause begins with a derivative of the word BY.

SORT CONNECTIVE	DESCRIPTION
BY	Orders output in ascending sequence.
BY-DSND	Orders output in descending sequence.
BY-EXP	Explodes multivalues and orders in ascending sequence. (This is covered later.)
BY-EXP-DSND	Explodes multivalued attributes and orders the output in ascending sequence.

Try the next sentence.

>SORT CUSTOMERS BY STATE COMPANY STATE CITY<r>

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause (The sequence clause which orders the output in state sequence.)
COMPANY STATE CITY	output-list

NOTE

The sequence clause specifies order only. It does not indicate output. Therefore, the columns COMPANY, STATE, and CITY must be explicitly requested.

```
>SORT CUSTOMERS BY STATE COMPANY STATE CITY<r>
```

```
PAGE 1
```

```
11:01:21 13 AUG 1988
```

```
CUSTOMERS. COMPANY..... STATE CITY.....
```

```
102      THE GUN STORE      CA      WEDONDO BEACH
103      AEROBIC DEN        CA      HOLLYWOOD
105      DIABOLICAL SOFTWARE CA      BYTELAND
106      PHINK, PHINK AND PHI CA      SANTA ANA
          NK
107      MONA LISA LTD      CA      VENICE BEACH
100      SHIPS R US         NY      NEW YORK
101      ACME CARROT        NY      BROOKLYN
104      SPEAK EASY         PA      GRAMMAR CITY
```

```
8 ITEMS LISTED.
```

Notice that the output is in STATE order, but random within each state.

Sorts can be sequenced. For example, the city can be sorted within the state.

```
>SORT CUSTOMERS BY STATE BY CITY COMPANY STATE CITY <r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE BY CITY	sequence clause (A two level sequence clause which orders the output in city sequence within state sequence.)
COMPANY STATE CITY	output-list

```
>SORT CUSTOMERS BY STATE BY CITY COMPANY STATE CITY<r>
```

```
PAGE 1
```

```
11:01:19 13 AUG 1988
```

```
CUSTOMERS. COMPANY..... STATE CITY.....
```

```
105      DIABOLICAL SOFTWARE CA      BYTELAND
103      AEROBIC DEN        CA      HOLLYWOOD
106      PHINK, PHINK AND PHI CA      SANTA ANA
          NK
107      MONA LISA LTD      CA      VENICE BEACH
102      THE GUN STORE      CA      WEDONDO BEACH
101      ACME CARROT        NY      BROOKLYN
100      SHIPS R US         NY      NEW YORK
104      SPEAK EASY         PA      GRAMMAR CITY
```

```
8 ITEMS LISTED.
```

Sorts can be tiered to as many levels as are required.

Try the previous examples using the BY-DSND clause modifier instead of BY. Notice how the output is ordered in descending order.

3.6 SELECTION CLAUSE

A selection clause restricts the output generated by ACCESS to only those items which pass the specified criteria.

3.6.1 Selection Clause Structure

The derivative of the word WITH, or some variation thereof, is used to begin a selection clause.

SELECT CONNECTIVE	DESCRIPTION
WITH criteria	Specifies the inclusion of all items which pass the criteria.
IF criteria	Same as WITH. IF is a synonym.
WITHOUT criteria	Specifies inclusion if the items do not pass the criteria.
WITH NO criteria	Same as WITHOUT.

The selection criteria can be a data definition name:

WITH BALANCE

to check for the existence of a BALANCE code. All items which have data in the BALANCE field are included. Items with a null BALANCE field are excluded.

You should have previously entered the DICT definition item name BALANCE. However, there is no data in the BALANCE attribute, (attribute 7), of any of the items.

```
>LIST CUSTOMERS WITH BALANCE COMPANY BALANCE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
WITH BALANCE	selection clause
COMPANY BALANCE	output-list

The result of which would be:

```
[401] NO ITEMS PRESENT
```

The selection criteria can also be a comparison of the contents of the attribute to a specified value. Review the following format:

WITH {datadef} {operator} {"value"}

The "datadef" is the data definition NAME, COMPANY, ADDRESS, etc.

The "operator" is a valid relational operator.

OPERATOR	DESCRIPTION
= or EQ	Equal to
# or NE	Not equal to
> or GT	Greater than
< or LT	Less than
>= or GE	Greater than or equal to
<= or LE	Less than or equal to

The "value" is any literal data string surrounded by double quotes ("").

The value string following the operator may be any series of letters and/or numbers that describes the criteria of the data you are selecting.

NOTE

The following rule must be followed when selecting your values:

Alphabetic values may be stored in upper or lowercase. Be sure the data matches case.

The following example generates a report of customers in the state of California, and shows the company name and state in the output.

```
>LIST CUSTOMERS WITH STATE = "CA" COMPANY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
WITH STATE = "CA"	selection clause (The selection clause which includes only items with a CA in the STATE attribute.
COMPANY STATE	output-list

or the alternative form:

```
>LIST CUSTOMERS WITH STATE "CA" COMPANY STATE<r>
```

Notice that the equal sign (=) is the only optional operator. Equal is implied if not specified.

```
>LIST CUSTOMERS WITH STATE = "CA" COMPANY STATE<r>
```

```
PAGE 1
```

```
08:06:04 15 AUG 1988
```

```
CUSTOMERS. COMPANY..... STATE
```

```
106      PHINK, PHINK AND PHI CA
          NK
107      MONA LISA LTD          CA
102      THE GUN STORE          CA
103      AEROBIC DEN            CA
105      DIABOLICAL SOFTWARE    CA
```

```
5 ITEMS LISTED.
```

The next example generates a report of customers in states other than California.

```
>SORT CUSTOMER BY STATE WITH STATE # "CA" COMPANY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMER	filename
BY STATE	sequence clause (The sequence clause which orders output by ascending state.)
WITH STATE # "CA"	selection clause (The selection clause which excludes items that have CA in the STATE attribute.)
COMPANY STATE	output-list

or in the alternative format:

```
>SORT CUSTOMER BY STATE WITHOUT STATE "CA" COMPANY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMER	filename
BY STATE	sequence clause
WITHOUT STATE "CA"	selection clause
COMPANY STATE	output-list

```
>SORT CUSTOMER BY STATE WITHOUT STATE "CA" COMPANY STATE<r>
```

```
PAGE 1
```

```
08:20:19 15 AUG 1988
```

```
CUSTOMERS. COMPANY..... STATE
```

```
100      SHIPS R US            NY
101      ACME CARROT           NY
104      SPEAK EASY             PA
```

```
3 ITEMS LISTED.
```

3.6.2

Multiple Selection Clauses

Multiple selection clauses can be specified by separating them with words called Logical Connectives (AND and OR).

```
>verb filename WITH criteria AND WITH criteria  
OR  
>verb filename WITH criteria OR WITH criteria
```

LOGICAL CONNECTIVE	DESCRIPTION
AND	Both criteria have to be true.
OR	Either or both criteria have to be true.

Here, the output is restricted to only those customers not in California and not in Pennsylvania. Notice that the word IF is being used in place of WITH. These words are synonyms, so it makes no difference. It's a grammatical choice.

```
>LIST CUSTOMERS IF STATE # "CA" AND IF STATE # "PA" COMPANY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
IF STATE # "CA" AND IF STATE # "PA"	selection clause
COMPANY STATE	output-list

```
>LIST CUSTOMERS IF STATE # "CA" AND IF STATE # "PA" COMPANY STATE<r>
```

```
PAGE 1 16:40:32 15 AUG 1988
```

```
CUSTOMERS. COMPANY..... STATE
```

```
100 SHIPS R US NY  
101 ACME CARROT NY
```

```
2 ITEMS LISTED.
```

Here, the output includes California and Pennsylvania, but excludes everything else:

```
>LIST CUSTOMERS WITH STATE = "CA" OR WITH STATE = "PA" COMPANY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
WITH STATE = "CA" OR WITH STATE = "PA"	selection clause
COMPANY STATE	output-list

```
>LIST CUSTOMERS WITH STATE = "CA" OR WITH STATE = "PA" COMPANY STATE<r>
```

PAGE 1

16:55:47 15 AUG 1988

CUSTOMERS. COMPANY..... STATE

```
106      PHINK, PHINK AND PHI CA
          NK
107      MONA LISA LTD          CA
102      THE GUN STORE          CA
103      AEROBIC DEN            CA
104      SPEAK EASY             PA
105      DIABOLICAL SOFTWARE    CA
```

6 ITEMS LISTED.

Using logical connectives requires the full WITH clause to be repeated. You can't say:

```
>LIST CUSTOMERS IF STATE # "CA" AND # "PA"<r>
```

You MUST say:

```
>LIST CUSTOMERS IF STATE # "CA" AND IF STATE # "PA"<r>
```

OR connectives can be implied if the value is being compared to the same field. The OR example can more easily be written:

```
>LIST CUSTOMERS WITH STATE = "CA" "PA" COMPANY STATE<r>
```

This lists the CUSTOMERS file as before, with the state as either California or Pennsylvania.

3.6.3

Value String Searching

When the EQ (equal to) is used, the system is looking for a character-for-character match. This sometimes presents a problem when only part of the field is being searched. You must use "string searching" when you want to find a particular string of characters in a Data Descriptor. Brackets ([]) must be used to indicate a string search.

For example, you are searching for a CONTACT in the CUSTOMERS file with the name "JOHNSON, BILL."

Here are several ways of searching for these people:

VALUE	RESULT
"JOHNSON, BILL"	These characters must be exactly matched. The system would not pull, for example, "JOHNSON, BILL R." or "JOHNSON, BILL JR." or even "JOHNSON, BILL."
"[BILL"	Using the left bracket before "BILL" tells the system that "BILL" may be preceded by any characters, as long as the string ends with the characters "BILL." Therefore, you would get "JOHNSON, BILL" or "SMITH, BILL" or "BUFFALO, BILL."
"JOHNSON]"	Using the right bracket after "JOHNSON" indicates that "JOHNSON" may be followed by any characters as long as it begins with "JOHNSON." Therefore, you get "JOHNSON, CARL" or "JOHNSON, BILL" or "JOHNSON, EDNA."
"[JOHNSON]"	Using both brackets tells the system that "JOHNSON" may be preceded or followed by any set of characters. Therefore, you get "JOHNSON, BILL" or "JOHNSON, BECKY" or "DAVIS, JOHNSON."

The following example outputs a report of all customers with an A in their company name.

```
>LIST CUSTOMERS WITH COMPANY = "[A]" COMPANY<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
WITH COMPANY = "[A]"	selection clause
COMPANY	output-list

```
>LIST CUSTOMERS WITH COMPANY = "[A]" COMPANY<r>
```

```
PAGE 1
```

```
08:24:25 16 AUG 1988
```

```
CUSTOMERS. COMPANY.....
```

```
106      PHINK, PHINK AND PHI
          NK
107      MONA LISA LTD
101      ACME CARROT
103      AEROBIC DEN
104      SPEAK EASY
105      DIABOLICAL SOFTWARE
```

```
6 ITEMS LISTED.
```

The following example generates a report of CUSTOMERS whose zip begins with a 9, in order of CONTACT name. Since California's zip codes begin with a 9, the report includes all customers in California.

```
>SORT CUSTOMERS BY CONTACT WITH ZIP = "9]" CONTACT COMPANY ZIP<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY CONTACT	sequence clause
WITH ZIP = "9]"	selection clause
CONTACT COMPANY ZIP	output-list

```
>SORT CUSTOMERS BY CONTACT WITH ZIP = "9]" CONTACT COMPANY ZIP<r>
```

```
PAGE 1
```

```
08:19:35 16 AUG 1988
```

```
CUSTOMERS. CONTACT..... COMPANY..... STATE ZIP.....
```

107	DAVINCI, LEONARDO	MONA LISA LTD	CA	97771
102	FUDD, ELMER	THE GUN STORE	CA	91234
105	GROG, BILL SPIDER	DIABOLICAL SOFTWARE	CA	90001
106	PHINK, CLYDE	PHINK, PHINK AND PHI	CA	90234
		NK		
103	PORCELANA, PORCHE	AEROBIC DEN	CA	91329

```
5 ITEMS LISTED.
```

3.6.4

Wildcard Searches

Wildcard searches allow the specification of a "don't care" character or series of characters in the value string. The wildcard character is represented by a caret (^).

```
>LIST CUSTOMERS WITH CONTACT = "[G^" CONTACT CITY<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
WITH CONTACT = "[G^"	selection clause (Include contacts which end in the two letter combination of a G followed by any character.)
CONTACT CITY	output-list

```

>LIST CUSTOMERS WITH CONTACT = "[G^" CONTACT CITY<r>

PAGE 1                                15:26:17  16 AUG 1988

CUSTOMERS: CONTACT..... CITY.....

101          BUNNY, BUGS          BROOKLYN

END OF LIST

```

Bugs is the only one who passes the criteria.

3.6.5

ID Selection Criteria

Selection criteria can also depend on the item-id. This can be indicated in one of two ways, implied and specific.

The implied item-id selection criteria follows the filename. It consists of an operator and value. The reference to the item-id is implied.

```
>verb filename operator 'value'
```

The following example outputs as a list of CONTACTS in the CUSTOMERS file whose customer-id begins with a 1.

```
>SORT CUSTOMERS = '1]' CONTACT<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
= '1]'	id selection
CONTACT	output-list

The value must be surrounded by single quotes (').

Generate a listing of customer COMPANY and CITY if the customer-id has a 5 anywhere in it.

```
>LIST CUSTOMERS = '[5]' COMPANY CITY<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
= '[5]'	id selection
COMPANY CITY	output-list

```
>LIST CUSTOMERS = '[5]' COMPANY CITY<r>
```

```
PAGE 1
```

```
13:03:18 16 AUG 1988
```

```
CUSTOMERS. COMPANY..... CITY.....
```

```
105          DIABOLICAL SOFTWARE BYTELAND
```

```
END OF LIST
```

Item-id selection can also be explicit. Remember the data definition that you entered entitled CUSTNUM. CUSTNUM points to attribute 0. Attribute 0 is the item-id.

The same two sentences in explicit format are:

```
>SORT CUSTOMERS WITH CUSTNUM = "[1]" CONTACT<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
WITH CUSTNUM = "[1]"	selection clause
CONTACT	output-list

```
>LIST CUSTOMERS WITH CUSTNUM = "[5]" COMPANY CITY<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
WITH CUSTNUM = "[5]"	selection clause
COMPANY CITY	output-list

An explicit selection clause requires that the double quotes (") surround the test value.

3.7

OUTPUT MODIFIERS AND OPTIONS

The ACCESS sentence may end in a series of modifiers used to change the default output characteristics. Output modifiers can be specified as words or as command sentence options.

The detailed use of each is demonstrated in the chapters, "Columnar Reporting," and "Mailing Labels."

The list of output modifiers and their related options is as follows:

OUTPUT MODIFIER	DESCRIPTION
ID-SUPP	Suppresses the automatic display of the item-id column. Also indicated with the (I) option.
DET-SUPP	Used in an ACCESS statement in which a SORT sequence is combined with a BREAK-ON modifier. (See Chapter 4, "Columnar Reporting.") Its function is to suppress the display of all the detail lines contributing to a break line, and, instead, to display the data value that caused the break to occur. Also indicated by the (D) option.
HDR-SUPP	Suppresses the default page heading that contains the time, date, and page number. Also indicated by the (H) option.
COL-HDR-SUPP	Suppresses both the default page heading which normally includes time, date and page number, as well as the date column headings. Also indicated by the (C) option.
NOPAGE	Causes the report to display on the CRT (default) without pausing for the user to enter a <RETURN> between each page of output. This is typically used when the report is being printed on a terminal attached (slave) printer. Also indicated by the (N) option.
LPTR	Directs the report output to the spooler currently assigned to the line printer. Also indicated by the (P) option.

Simple example of output modifier use:

The following sentence prints an ordered report of the CUSTOMERS file.

```
>SORT CUSTOMERS BY STATE BY COMPANY STATE COMPANY CONTACT LPTR<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE BY COMPANY	sequence clause
STATE COMPANY CONTACT	output-list
LPTR	output modifier

```
>SORT CUSTOMERS BY STATE BY COMPANY STATE COMPANY CONTACT LPTR<r>
ENTRY #1
```

The spooler entry number.

The printed output is:

PAGE 1	08:10:03 16 AUG 1988
CUSTOMERS. STATE COMPANY..... CONTACT.....	
103	CA AEROBIC DEN PORCELANA, PORCHE
105	CA DIABOLICAL SOFTWARE GROG, BILL SPIDER
107	CA MONA LISA LTD DAVINCI, LEONARDO
106	CA PHINK, PHINK AND PHI PHINK, CLYD NK
102	CA THE GUN STORE FUDD, ELMER
101	NY ACME CARROT BUNNY, BUGS
100	NY SHIPS R US HORNBLLOWER, HORATIO
104	PA SPEAK EASY PARTICIPLE, DANGLY
8 ITEMS LISTED.	

An equivalent sentence that outputs to the printer:

```
>SORT CUSTOMERS BY STATE BY COMPANY STATE COMPANY CONTACT (P)<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE BY COMPANY	selection clause
STATE COMPANY CONTACT	output-list
(P)	option

3.8 THROWAWAY CONNECTIVES

Throwaways are words which are not processed in the ACCESS sentence, but can be used to "round out" the grammar of the sentence. The following are the available throwaways:

A	AN	ARE	ANY	FILE
FOR	IN	ITEMS	OF	THE

Using these throwaway connectives, an ACCESS sentence can be entered as follows:

```
>SORT THE CUSTOMERS FILE BY STATE COMPANY STATE<r>
```



throwaways

New throwaways can be created by copying the existing words to new names. Since these connectives reside in the master dictionary of an account, new throwaways are created by using the COPY verb and the MD file.

```
>COPY MD A<r>
TO: SHOWING<r>

1 A TO SHOWING
1 ITEM COPIED
```

The previous statement can be re-entered as:

```
>SORT THE CUSTOMERS FILE BY THE STATE SHOWING THE COMPANY STATE<r>
```



throwaways

```
>SORT THE CUSTOMERS FILE BY THE STATE SHOWING THE COMPANY STATE<r>
```

PAGE 1

10:10:00 08 DEC 1988

CUSTOMERS. COMPANY..... STATE

102	THE GUN STORE	CA
103	AEROBIC DEN	CA
105	DIABOLICAL SOFTWARE	CA
106	PHINK, PHINK AND PHI	CA
	NK	
107	MONA LISA LTD	CA
100	SHIPS R US	NY
101	ACME CARROT	NY
104	SPEAK EASY	PA

8 ITEMS LISTED.

EXERCISE**3**

1. The basic components of an ACCESS command sentence are _____ and _____.
2. Name four commonly used ACCESS verbs.

3. Verbs travel in pairs. One always generates a _____ output, while the other generates a _____ output.
4. Which file levels do the following command sentences report?

>LIST CUSTOMERS _____

>SORT DICT CUSTOMERS _____

>LIST ONLY CUSTOMERS _____

>SORT DATA CUSTOMERS _____
5. The ACCESS verb which outputs only a count of items is _____.
6. The first column of a LIST or SORT output is made up of _____.
7. True or False. The following ACCESS sentence lists only three specific items.

>LIST CUSTOMERS '102''103''106'

8. Write an ACCESS command sentence which outputs the company name, address and city for all entries in the CUSTOMERS file.

> _____

9. Write an ACCESS sentence which outputs the same information as above, but reports only CUSTOMERS 100 and 101.

> _____

10. Match the following connectives:

_____ BY	a. Beginning of a selection clause. Includes items which pass the criteria.
_____ WITHOUT	b. A throwaway connective.
_____ BY-EXP	c. Sorts in ascending sequence.
_____ IF	d. A selection connective indicating items which do not pass the selection criteria.
_____ AND	e. Sorts in descending sequence.
_____ BY-DSND	f. A logical connective meaning both selection criteria must be true.
_____ WITH	g. A synonym for WITH.
_____ THE	h. Another throwaway.
_____ A	i. A logical connective meaning either selection criteria can be true.
_____ OR	j. Explodes multivalued attributes for sorting.

11. Write the command sentence to sort the CUSTOMERS file by contact name within state and show the contact name and state abbreviation.

> _____

12. Match the selection clause:

- | | |
|---------------------------------|--|
| _____ WITH STATE | a. Include customers in either California or Texas. |
| _____ WITH BALANCE GE "1000.00" | b. Include only those items that have a null address attribute. |
| _____ IF CONTACT = "S]" | c. Include customers in the 213 area code. |
| _____ WITHOUT ADDRESS | d. Exclude customers with no contact name. |
| _____ WITH STATE "CA""TX" | e. Include those customers with a balance due greater than or equal to 1000 dollars. |
| _____ WITH NO CONTACT | f. Include customers whose contact names start with an S. |
| _____ IF PHONE = "213]" | g. Include only those customers with a non-null state attribute. |

13. The wildcard search character is a _____.

14. Write a selection clause for the CUSTOMERS file that includes those company names whose second to the last character is a K.

15. Write a full ACCESS statement which sorts the output by the balance due for only those customers in California in the CUSTOMERS file. The report should output both fields.

> _____

16. In the following ACCESS sentence, match each description with the correct portion of the sentence.

>SORT CUSTOMERS BY STATE BY ZIP WITH BALANCE COMPANY STATE ZIP BALANCE LPTR

- | | | |
|-------|--------------------|---------------------|
| _____ | a. output-list | d. filename |
| _____ | b. verb | e. selection clause |
| _____ | c. output modifier | f. sequence clause |



4

COLUMNAR REPORTING

This chapter demonstrates the techniques for formatting a columnar report. Please review Chapter 3, "The ACCESS Command Sentence," to become familiar with the command sentence components that are used in these report examples.

Also, make sure that your terminal characteristics are set correctly before continuing. Setting up terminal and printer characteristics is discussed at length later in this chapter in Section 4.8, "Choosing the Output Device." For now, just enter the following:

```
>TERM 79,23
```

This says to set the terminal width and depth to 79 columns and 23 lines.

4.1

THE BASICS

Columnar reports are generated by the ACCESS verbs LIST and SORT.

VERB	DESCRIPTION
LIST	Output a columnar report in random order.
SORT	Output a columnar report in sorted order of item-ids if no BY clause is used, or BY any specified data definition.

The general form is:

```
>LIST filename<r>
```

or

```
>SORT filename<r>
```

For example:

```
>LIST CUSTOMERS<r>
```

```
PAGE 1
```

```
16:20:15 12 AUG 1988
```

```
CUSTOMERS.
```

```
106  
107  
100  
101  
102  
103  
104  
105
```

```
8 ITEMS LISTED.
```

SORT generates a listing of the item-ids in alphabetical or numerical sequence.

```
>SORT CUSTOMERS<r>
```

```
PAGE 1
```

```
08:13:39 20 FEB 1987
```

```
CUSTOMERS.
```

```
100  
101  
102  
103  
104  
105  
106  
107
```

```
8 ITEMS LISTED.
```

A SORT by company name is specified as follows:

```
>SORT CUSTOMERS BY COMPANY<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY COMPANY	sequence clause

```
>SORT CUSTOMERS BY COMPANY<r>
```

```
PAGE 1
```

```
12:52:14 08 SEP 1988
```

```
CUSTOMERS.
```

```
101  
103  
105  
107  
106  
100  
104  
102
```

```
8 ITEMS LISTED.
```

Notice that even though the sequence clause is used, the report still only shows the item-ids. Read on to see how columnar output can be requested.

4.2

EXPLICIT OUTPUT FORMATTING

The previous ACCESS sentences haven't been much help in viewing any data other than the item-ids. Here's where the data definition items come into play. Output data columns are specified in the following form:

```
>verb filename name1 name2 name3 ....
```

Each definition name generates a column of output.

```
>LIST CUSTOMERS COMPANY CONTACT CITY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
COMPANY CONTACT CITY STATE	output-list

```
>LIST CUSTOMERS COMPANY CONTACT CITY STATE<r>
```

```
PAGE 1
```

```
09:54:24 13 AUG 1988
```

CUSTOMERS.	COMPANY.....	CONTACT.....	CITY.....	STATE
106	PHINK, PHINK AND PHI NK	PHINK, CLYDE	SANTA ANA	CA
107	MONA LISA LTD	DAVINCI, LEONARDO	VENICE BEACH	CA
100	SHIPS R US	HORNBLLOWER, HORATIO	NEW YORK	NY
101	ACME CARROT	BUNNY, BUGS	BROOKLYN	NY
102	THE GUN STORE	FUDD, ELMER	WEDONDO BEACH	CA
103	AEROBIC DEN	PORCELANA, PORCHE	HOLLYWOOD	CA
104	SPEAK EASY	PARTICIPLE, DANGLY	GRAMMAR CITY	PA
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	BYTELAND	CA

```
8 ITEMS LISTED.
```

Notice that company 106 has a name wider than the current column width. The system automatically wrapped the data on to the next line in the same column.

Let's take a look at the same report using the SORT verb.

```
>SORT CUSTOMERS COMPANY CONTACT CITY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
COMPANY CONTACT CITY STATE	output-list

>SORT CUSTOMERS COMPANY CONTACT CITY STATE<r>

PAGE 1

09:59:02 13 AUG 1988

CUSTOMERS.	COMPANY.....	CONTACT.....	CITY.....	STATE
100	SHIPS R US	HORNBLOWER, HORATIO	NEW YORK	NY
101	ACME CARROT	BUNNY, BUGS	BROOKLYN	NY
102	THE GUN STORE	FUDD, ELMER	WEDONDO BEACH	CA
103	AEROBIC DEN	PORCELANA, PORCHE	HOLLYWOOD	CA
104	SPEAK EASY	PARTICIPLE, DANGLY	GRAMMAR CITY	PA
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	BYTELAND	CA
106	PHINK, PHINK AND PHI NK	PHINK, CLYDE	SANTA ANA	CA
107	MONA LISA LTD	DAVINCI, LEONARDO	VENICE BEACH	CA

8 ITEMS LISTED.

The SORT verb orders the output by the item-ids.

Now, here is the same SORT sentence with the output in COMPANY order.

>SORT CUSTOMERS BY COMPANY COMPANY CONTACT CITY STATE<r>

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY COMPANY	sequence clause
COMPANY CONTACT CITY STATE	output-list

>SORT CUSTOMERS BY COMPANY COMPANY CONTACT CITY STATE<r>

PAGE 1

13:05:51 08 SEP 1988

CUSTOMERS.	COMPANY.....	CONTACT.....	CITY.....	STATE
101	ACME CARROT	BUNNY, BUGS	BROOKLYN	NY
103	AEROBIC DEN	PORCELANA, PORCHE	HOLLYWOOD	CA
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	BYTELAND	CA
107	MONA LISA LTD	DAVINCI, LEONARDO	VENICE BEACH	CA
106	PHINK, PHINK AND PHI NK	PHINK, CLYDE	SANTA ANA	CA
100	SHIPS R US	HORNBLOWER, HORATIO	NEW YORK	NY
104	SPEAK EASY	PARTICIPLE, DANGLY	GRAMMAR CITY	PA
102	THE GUN STORE	FUDD, ELMER	WEDONDO BEACH	CA

8 ITEMS LISTED.

Notice that the data definition COMPANY must be repeated in the output-list as well as within the sequence clause. The sequence clause does not indicate output, only order.

IMPLICIT OUTPUT FORMATTING

Files can be given a group of default columns to be output whenever a simple ACCESS statement is entered. This is called implicit formatting since the columns are implied, not explicitly specified. All you have to do is to enter:

```
>LIST filename
or
>SORT filename
```

and a default report format will display.

Implicit columns are set up by creating special data definitions in the dictionary of the file. These definitions are special because they are not given names. They are given sequential numbers like 1, 2, 3, etc. Each number represents the default columnar position in the implicit report.

We're going to make the previous explicit example work implicitly.

If you wish, review the COPY command in your system manuals before continuing. Now, enter the following:

```
>COPY DICT CUSTOMERS COMPANY CONTACT CITY STATE<r>
TO:1 2 3 4<r>
```

The system wants to know
where these are being copied.

This procedure copies the DICT item COMPANY to an item called 1, CONTACT to 2, CITY to 3, and STATE to 4.

Here's what they look like.

ID:	1	2	3	4
001	A	A	A	A
002	1	2	4	5
003				
004				
005				
006				
007				
008				
009	L	L	L	L
010	20	20	15	2

The numbered names of these dictionary definitions represent the column, NOT THE ATTRIBUTE. Notice how column 4 is used to point to attribute 5.

Now, enter this simple command and see what happens.

```
>SORT CUSTOMERS<r>
```

```
PAGE 1
```

```
13:13:35 08 SEP 1988
```

```
CUSTOMERS. 1..... 2..... 3..... 4.
```

100	SHIPS R US	HORNBLOWER, HORATIO	NEW YORK	NY
101	ACME CARROT	BUNNY, BUGS	BROOKLYN	NY
102	THE GUN STORE	FUDD, ELMER	WEDONDO BEACH	CA
103	AEROBIC DEN	PORCELANA, PORCHE	HOLLYWOOD	CA
104	SPEAK EASY	PARTICIPLE, DANGLY	GRAMMAR CITY	PA
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	BYTELAND	CA
106	PHINK, PHINK AND PHI	PHINK, CLYDE	SANTA ANA	CA
	NK			
107	MONA LISA LTD	DAVINCI, LEONARDO	VENICE BEACH	CA

```
8 ITEMS LISTED.
```

The four columns defined by data definitions 1, 2, 3, and 4, automatically are displayed by explicitly requesting them in an output-list. The column headers default to the item-ids of the data definitions.

The column headers can be changed by using attribute 3 of the data definitions. Enter the following:

```
>ED DICT CUSTOMERS 1 2 3 4<r>
```

```
1
```

```
Top
```

```
.3<r>
```

```
003
```

```
.R//Company<r>
```

```
003 Company
```

```
.FI<r>
```

```
'1' FILED
```

```
2
```

```
Top
```

```
.3<r>
```

```
003
```

```
.R//Contact<r>
```

```
003 Contact
```

```
.FI<r>
```

```
'2' FILED
```

```
3
```

```
Top
```

```
.3<r>
```

```
003
```

```
.R//City<r>
```

```
003 City
```

```
.FI<r>
```

```
'3' FILED
```

```
4
```

```
Top
```

```
.3<r>
```

```
003
```

```
.R//State<r>
```

```
003 State
```

```
.FI<r>
```

```
'3' FILED
```

Edit all 4 data definitions.

Starts with Item '1'.

Goto attribute 3.

Replace the first occurrence of a null with the word company.

File it.

Continue with Item '2'.

Now enter the same sentence and review the results.

>SORT CUSTOMERS<r>

PAGE 1

13:13:35 08 SEP 1988

CUSTOMERS.	COMPANY.....	CONTACT.....	CITY.....	STATE
100	SHIPS R US	HORNBLOWER, HORATIO	NEW YORK	NY
101	ACME CARROT	BUNNY, BUGS	BROOKLYN	NY
102	THE GUN STORE	FUDD, ELMER	WEDONDO BEACH	CA
103	AEROBIC DEN	PORCELANA, PORCHE	HOLLYWOOD	CA
104	SPEAK EASY	PARTICIPLE, DANGLY	GRAMMAR CITY	PA
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	BYTELAND	CA
106	PHINK, PHINK AND PHI NK	PHINK, CLYDE	SANTA ANA	CA
107	MONA LISA LTD	DAVINCI, LEONARDO	VENICE BEACH	CA

8 ITEMS LISTED.

Now try this:

>SORT CUSTOMERS BY COMPANY<r>

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY COMPANY	sequence clause

>SORT CUSTOMERS BY COMPANY<r>

PAGE 1

13:27:32 08 SEP 1988

CUSTOMERS.	COMPANY.....	CONTACT.....	CITY.....	STATE
101	ACME CARROT	BUNNY, BUGS	BROOKLYN	NY
103	AEROBIC DEN	PORCELANA, PORCHE	HOLLYWOOD	CA
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	BYTELAND	CA
107	MONA LISA LTD	DAVINCI, LEONARDO	VENICE BEACH	CA
106	PHINK, PHINK AND PHI NK	PHINK, CLYDE	SANTA ANA	CA
100	SHIPS R US	HORNBLOWER, HORATIO	NEW YORK	NY
104	SPEAK EASY	PARTICIPLE, DANGLY	GRAMMAR CITY	PA
102	THE GUN STORE	FUDD, ELMER	WEDONDO BEACH	CA

8 ITEMS LISTED.

Again, no output-list is required.

Implicit formatting may not be what is required for all circumstances. It can be deactivated by using an explicit output-list.

```
>LIST CUSTOMERS COMPANY CONTACT<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
COMPANY CONTACT	output-list

Or, implicit formatting can be deactivated by using the ONLY modifier preceding the filename.

```
>LIST ONLY CUSTOMERS<r>
```

Try these commands and note the results.

4.4 HANDLING MULTIVALUED OUTPUT FORMATTING

Certain attributes in an item may contain more than a single data value. Multivalued attributes are used to hold lists of a like data. For example, an invoice item may have an attribute which is a list of product numbers, followed by an attribute which is a list of quantities sold. In our sample CUSTOMERS file, the PHONE attribute contains multiple phone numbers.

This section shows how multivalued attributes are reported and sorted.

4.4.1 *Simple Output*

Whenever a multivalue field is output in a columnar report, values are stacked within the column in the order that they are stored in the attribute.

Enter the following command sentence:

```
>LIST CUSTOMERS COMPANY CONTACT PHONE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
COMPANY CONTACT PHONE	output-list

The output is as follows:

```
>LIST CUSTOMERS COMPANY CONTACT PHONE<r>
```

```
PAGE 1
```

```
09:30:15 08 DEC 1988
```

```
CUSTOMERS. COMPANY..... CONTACT..... PHONE.....
```

100	SHIPS R US	HORNBLOWER, HORATIO	2125557676
			2125556666
101	ACME CARROT	BUNNY, BUGS	2125556666
			2125556661
102	THE GUN STORE	FUDD, ELMER	2135559999
103	AEROBIC DEN	PORCELANA, PORCHE	2135557777
104	SPEAK EASY	PARTICIPLE, DANGLY	2155554433
			2155556321
			2155554444
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	4085554444
106	PHINK, PHINK AND PHI	PHINK, CLYDE	2135551112
	NK		
107	MONA LISA LTD	DAVINCI, LEONARDO	2135554444
			2135554321

```
8 ITEMS LISTED.
```

Don't let the extra space generated for CUSTOMERS 106 fool you. It is only a single value. The extra line is generated because the company name wraps to the next line. This may be annoying, but live with it for now. It will be taken care of in a later chapter.

4.4.2

Sorted Output

Sorting by multivalued attributes can create some difficulty since the BY clause uses entire attributes as one string for sorting. To understand this a little better, enter the following sentence:

```
>SORT CUSTOMERS BY PHONE COMPANY CONTACT PHONE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY PHONE	sequence clause
COMPANY CONTACT PHONE	output-list

```
>SORT CUSTOMERS BY PHONE COMPANY CONTACT PHONE<r>
```

```
PAGE 1
```

```
09:30:57 08 DEC 1988
```

```
CUSTOMERS. COMPANY..... CONTACT..... PHONE.....
```

101	ACME CARROT	BUNNY, BUGS	2125556666
			2125556661
100	SHIPS R US	HORNBLOWER, HORATIO	2125557676
			2125556666
106	PHINK, PHINK AND PHI	PHINK, CLYDE	2135551112
	NK		
107	MONA LISA LTD	DAVINCI, LEONARDO	2135554444
			2135554321
103	AEROBIC DEN	PORCELANA, PORCHE	2135557777
102	THE GUN STORE	FUDD, ELMER	2135559999
104	SPEAK EASY	PARTICIPLE, DANGLY	2155554433
			2155556321
			2155554444
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	4085554444

```
8 ITEMS LISTED.
```

Notice the PHONE column. The individual phone numbers are not in sorted order. This is the reason that the BY-EXP modifier must be used. BY-EXP breaks up the value lists so that they can be reported in the correct sequence.

Enter the following command sentence:

```
>SORT CUSTOMERS BY-EXP PHONE COMPANY CONTACT PHONE<r>
```

SENTENCE ELEMENTS	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY-EXP PHONE	sequence clause
COMPANY CONTACT PHONE	output-list

```
>SORT CUSTOMERS BY-EXP PHONE COMPANY CONTACT PHONE<r>
```

```
PAGE 1                                09:31:22  08 DEC 1988
```

CUSTOMERS.	COMPANY.....	CONTACT.....	PHONE.....
101	ACME CARROT	BUNNY, BUGS	2125556661
100	SHIPS R US	HORNBLOWER, HORATIO	2125556666
101	ACME CARROT	BUNNY, BUGS	2125556666
100	SHIPS R US	HORNBLOWER, HORATIO	2125557676
106	PHINK, PHINK AND PHI	PHINK, CLYDE	2135551112
	NK		
107	MONA LISA LTD	DAVINCI, LEONARDO	2135554321
107	MONA LISA LTD	DAVINCI, LEONARDO	2135554444
103	AEROBIC DEN	PORCELANA, PORCHE	2135557777
102	THE GUN STORE	FUDD, ELMER	2135559999
104	SPEAK EASY	PARTICIPLE, DANGLY	2155554433
104	SPEAK EASY	PARTICIPLE, DANGLY	2155554444
104	SPEAK EASY	PARTICIPLE, DANGLY	2155556321
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER	4085554444

13 ITEMS LISTED.

The phone numbers are now in correctly sorted order. Note that any item which contains more than one value for the phone number is output in an equivalent number of lines in the report. For example, CUSTOMER 101 has two values in the PHONE field and 101 appears twice in the report. This explains why 13 items are listed even though there are only 8 items in the entire file.

4.5

CUSTOM HEADINGS AND FOOTINGS

The system automatically places a heading on columnar reports. This default heading consists of the system time, date and current page number. However, you have the option to override the default heading and specify a custom report heading and footing by using the following modifiers.

MODIFIER	DESCRIPTION
HEADING	Specifies that the user defined text is to be output at the top of each output page.
FOOTING	Specifies that the user defined text is to be output at the bottom of each page.

HEADING and FOOTING Text:

Each of these modifiers must be followed by a literal string specifying the actual text to be used in the Heading or Footing. The general form of the HEADING or FOOTING clause is:

HEADING 'text... 'options' ..text... 'option''

or

FOOTING "text... 'options' ..text... 'option'"

System level options may be included within the text. These options must be set apart by surrounding them with single quote (') marks.

The following is a list of options:

OPTION	DESCRIPTION
C	Centers output.
B	Insert the value causing the report break. Must be used in conjunction with the BREAK-ON modifier options. See Section 4.6 "Report Breaks" for more information.
D	Outputs the System Date.
F	Outputs the Filename.
Fn	Outputs the Filename left justified in a field of 'n' blanks.
I	Outputs Item Name.
In	Outputs Item Name left justified in a field of 'n' blanks.
L	Outputs a linefeed and carriage return, skips a line.
P	Outputs page number, right justified in a field of four blanks.
PN	Outputs page number, left justified.
T	Outputs time and date.
"	Outputs one single quote. Since single quotes are used to set off options, this allows a single quote to be a part of the text.

Here is a simple use of the HEADING modifier.

```
>SORT CUSTOMERS BY COMPANY COMPANY CONTACT HEADING "COMPANY LIST 'CL' "<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY COMPANY	sequence clause
COMPANY CONTACT	output-list
HEADING "COMPANY LIST 'CL' "	heading clause

The heading clause is:

"COMPANY LIST 'CL' "

This contains the literal string (text) COMPANY LIST modified by the options, C, center the text on this line, and L, force a line feed. The centering is based on the width of the chosen output device, in this case, the terminal. Notice that multiple options can be specified within a single set of single quotes.

Column numbers are provided here to make the positioning of the heading more apparent. These do not appear on the report. The output is as follows:

>SORT CUSTOMERS BY COMPANY COMPANY CONTACT HEADING "COMPANY LIST\CL\' "<r>

1 2 3 4 5 6 7
12345678901234567890123456789012345678901234567890123456789

COMPANY LIST

Heading.

CUSTOMERS. COMPANY..... CONTACT.....

101	ACME CARROT	BUNNY, BUGS
103	AEROBIC DEN	PORCELANA, PORCHE
105	DIABOLICAL SOFTWARE	GROG, BILL SPIDER
107	MONA LISA LTD	DAVINCI, LEONARDO
106	PHINK, PHINK AND PHI	PHINK, CLYDE
	NK	
100	SHIPS R US	HORNBLOWER, HORATIO
104	SPEAK EASY	PARTICIPLE, DANGLY
102	THE GUN STORE	FUDD, ELMER

Headings and footings can be used together or separately. Here is an example of both the HEAD-ING and FOOTING modifiers in the same command sentence.

>LIST CUSTOMERS CONTACT COMPANY HEADING "CONTACTS\LDLL\' "FOOTING" \'L\' PAGE \'PN\' "<r>

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
CONTACT COMPANY	output-list
HEADING "CONTACTS\LDLL\'"	heading clause
FOOTING " \'L\' PAGE \'PN\'"	footing clause

The HEADING text specifies the literal string "CONTACTS" along with a line feed, followed by the current system date, followed by two more line feeds.

"CONTACTS\LDLL\'"

The FOOTING text specifies a line feed, followed by the literal string "PAGE" and the current page number left justified.

" \'L\' PAGE \'PN\'"

>LIST CUSTOMERS CONTACT COMPANY HEADING "CONTACTS\LDLL\' "FOOTING" \'L\' PAGE \'PN\' "<r>

CONTACTS
06 DEC 1988

Heading.

CUSTOMERS. CONTACT..... COMPANY.....

106	PHINK, CLYDE	PHINK, PHINK AND PHI
		NK
107	DAVINCI, LEONARDO	MONA LISA LTD
100	HORNBLOWER, HORATIO	SHIPS R US
101	BUNNY, BUGS	ACME CARROT
102	FUDD, ELMER	THE GUN STORE
103	PORCELANA, PORCHE	AEROBIC DEN
104	PARTICIPLE, DANGLY	SPEAK EASY
105	GROG, BILL SPIDER	DIABOLICAL SOFTWARE

PAGE 1

Footing.

4.6

GENERATING REPORT BREAKS

In a SORTed report, the output columns normally run together without a visual break.

```
>SORT CUSTOMERS BY STATE STATE COMPANY CITY<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
STATE COMPANY CITY	output-list

```
>SORT CUSTOMERS BY STATE STATE COMPANY CITY<r>
```

PAGE 1

11:18:55 06 DEC 1988

CUSTOMERS. STATE COMPANY..... CITY.....

102	CA	THE GUN STORE	WEDONDO BEACH
103	CA	AEROBIC DEN	HOLLYWOOD
105	CA	DIABOLICAL SOFTWARE	BYTELAND
106	CA	PHINK, PHINK AND PHI	SANTA ANA
		NK	
107	CA	MONA LISA LTD	VENICE BEACH
100	NY	SHIPS R US	NEW YORK
101	NY	ACME CARROT	BROOKLYN
104	PA	SPEAK EASY	GRAMMAR CITY

8 ITEMS LISTED.

The report break modifier allows a visual break as sorted values change.

MODIFIER	DESCRIPTION
BREAK-ON	Used to generate a visual "break" when a change occurs in a sorted column. The visual portion of the break is displayed as the string "****" between two blank lines on the report. If the TOTAL modifier is also in effect during a "break-on," the subtotal for each Data Descriptor specified with the TOTAL modifier is displayed at the break point.
TOTAL	Produces a columnar total of specified numeric Data Descriptor. Subtotals can be obtained by using the BREAK-ON modifier.

4.6.1

Simple Breaks

Enter the following sentence:

```
>SORT CUSTOMERS BY STATE BY COMPANY BREAK-ON STATE COMPANY CITY<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE BY COMPANY	sequence clause
BREAK-ON STATE COMPANY CITY	output-list

OUTPUT-LIST	DESCRIPTION
BREAK-ON STATE	Generates a report break as the STATE changes.
COMPANY CITY	Outputs the COMPANY and CITY columns.

Produces the output:

```

>SORT CUSTOMERS BY STATE BY COMPANY BREAK-ON STATE COMPANY CITY<r>
PAGE 1                                14:30:39  05 OCT 1988
CUSTOMERS. STATE COMPANY..... CITY.....

103      CA  AEROBIC DEN      HOLLYWOOD
105      CA  DIABOLICAL SOFTWARE BYTELAND
107      CA  MONA LISA LTD    VENICE BEACH
106      CA  PHINK, PHINK AND PHI SANTA ANA
          NK
102      CA  THE GUN STORE    WEDONDO BEACH
          ***
101      NY  ACME CARROT      BROOKLYN
100      NY  SHIPS R US       NEW YORK
          ***
104      PA  SPEAK EASY       GRAMMAR CITY
          ***
          ***
8 ITEMS LISTED.

```

Notice the summary line generates three asterisks (***) at the end of the report.

Breaks can be indicated at multiple levels. Try the following sentence.

```
>SORT CUSTOMERS BY STATE BY CITY BREAK-ON STATE BREAK-ON CITY<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE BY CITY	sequence clause
BREAK-ON STATE BREAK-ON CITY	output-list

>SORT CUSTOMERS BY STATE BY CITY BREAK-ON STATE BREAK-ON CITY<r>

PAGE 1

14:36:25 05 OCT 1988

CUSTOMERS. STATE CITY.....

105 CA BYTELAND

Second level
BREAK.

103 CA HOLLYWOOD

Second level
BREAK.

106 CA SANTA ANA

107 CA VENICE BEACH

102 CA WEDONDO BEACH

First level
BREAK.

101 NY BROOKLYN

100 NY NEW YORK

104 PA GRAMMAR CITY

8 ITEMS LISTED.

This report breaks at the second level for each line since the city changes for each line.

4.6.2

Totals and Sub-Totals

The TOTAL output modifier generates a grand total at the end of the report. (Make sure that you are totaling numeric fields, otherwise the result is zero.) TOTAL also automatically generates a sub-total for each break level.

Enter the following sentence:

>SORT CUSTOMERS BY STATE BREAK-ON STATE COMPANY TOTAL BALANCE<r>

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
BREAK-ON STATE COMPANY...	
...TOTAL BALANCE	output-list

OUTPUT-LIST	DESCRIPTION
BREAK-ON STATE	Generates a report break as the state changes.
COMPANY	Outputs the COMPANY name.
TOTAL BALANCE	Generates the sum of the BALANCE column and a sub-total as the STATE changes.

>SORT CUSTOMERS BY STATE BREAK-ON STATE COMPANY TOTAL BALANCE<r>

PAGE 1

14:42:20 05 OCT 1988

CUSTOMERS. STATE COMPANY..... BALANCE DUE.

102	CA	THE GUN STORE	12.34
103	CA	AEROBIC DEN	89.12
105	CA	DIABOLICAL SOFTWARE	4.56
106	CA	PHINK, PHINK AND PHI NK	126.75
107	CA	MONA LISA LTD	789.32

*** 1022.09

Subtotal for CA.

100	NY	SHIPS R US	567.80
101	NY	ACME CARROT	56.75

*** 624.55

Subtotal for NY.

104	PA	SPEAK EASY	23.42
-----	----	------------	-------

*** 23.42

Subtotal for PA.

*** 1670.06

Grand Total.

8 ITEMS LISTED.

4.6.3

Grand Totals

The GRAND-TOTAL modifier is used to generate text in place of the three asterisks (***) on the final report summary line. The general form is as follows:

GRAND-TOTAL "literal string":

```
>SORT CUSTOMERS BY STATE BREAK-ON STATE COMPANY...  
...TOTAL BALANCE GRAND-TOTAL "FINAL BALANCE"<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
BREAK-ON STATE COMPANY...	
...TOTAL BALANCE	output-list
GRAND-TOTAL "FINAL BALANCE"	grand total label

```
>SORT CUSTOMERS BY STATE BREAK-ON STATE COMPANY TOTAL BALANCE GRAND-TOTAL "FINAL BALANCE"<r>
```

PAGE 1 11:29:46 06 DEC 1988

CUSTOMERS. STATE COMPANY..... BALANCE DUE.

102	CA	THE GUN STORE	12.34
103	CA	AEROBIC DEN	89.12
105	CA	DIABOLICAL SOFTWARE	4.56
106	CA	PHINK, PHINK AND PHI NK	126.75
107	CA	MONA LISA LTD	789.32
	***		1022.09
100	NY	SHIPS R US	567.80
101	NY	ACME CARROT	56.75
	***		624.55
104	PA	SPEAK EASY	23.42
	***		23.42
		FINAL BALANCE	1670.06
		8 ITEMS LISTED.	

4.6.4

BREAK Options

The BREAK-ON modifier also has an option string which is used in the same way that the HEADING and FOOTING modifiers use options. The general format is as follows:

```
BREAK-ON dataname "text.. 'options' text.. 'options'"
```

The entire option string must be surrounded by double (") quotes, and the options must be surrounded by single (') quotes. A list of the options follows.

OPTION	DESCRIPTION
B	Designates that the value of the break is to be placed in the heading or footing, if the HEADING or FOOTING text string contains another 'B' option.
D	This suppresses the break line if there is only a single detail to report.
L	Break lines normally output a blank line before reporting the break. This suppresses the extra line.
N	Resets the page counter to one.
P	Forces a new page at the end of the break line.
R	Forces one or more break lines to the bottom of the same page, rather than going to a new page.
U	Underlines TOTAL columns.
V	Prints the value of the break at the break line rather than the three asterisks (***).
"	Places a literal single quote in the break text.

Re-enter the previous example using break options as follows:

```
>SORT CUSTOMERS BY STATE BREAK-ON STATE "TOT 'LU' " COMPANY TOTAL BALANCE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
BREAK-ON STATE "TOT 'LU' "...	
...COMPANY TOTAL BALANCE	output-list

```
>SORT CUSTOMERS BY STATE BREAK-ON STATE "TOT 'LU' " COMPANY TOTAL BALANCE<r>
```

PAGE 1 11:29:46 06 DEC 1988

CUSTOMERS. STATE COMPANY..... BALANCE DUE.

102	CA	THE GUN STORE	12.34
103	CA	AEROBIC DEN	89.12
105	CA	DIABOLICAL SOFTWARE	4.56
106	CA	PHINK, PHINK AND PHI NK	126.75
107	CA	MONA LISA LTD	789.32
	TOT		1022.09
100	NY	SHIPS R US	567.80
101	NY	ACME CARROT	56.75
	TOT		624.55
104	PA	SPEAK EASY	23.42
	TOT		23.42
***			1670.06

Underlines total columns.

Skip line is suppressed.

TOT is the break literal text.

8 ITEMS LISTED.

The BREAK-ON option string is as follows:

BREAK-ON STATE "TOT 'LU'"

Generates a report break when the STATE changes. At the break line print the text TOT, suppress the blank line, and underline the TOTAL fields.

Here is the same command sentence with a few more modifications. The break options indicate that each STATE is to be reported on a new page with the value of the break in the heading.

Enter the following command sentence:

```
>SORT CUSTOMERS BY STATE BREAK-ON STATE "TOT 'LUBP'" COMPANY...  
...TOTAL BALANCE HEADING "BALANCE FOR 'B' ON 'DL' PAGE 'PNL' "<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
BREAK-ON STATE "TOT 'LUBP'" ...	
...COMPANY TOTAL BALANCE	output-list
HEADING "BALANCE FOR 'B' ...	
...ON 'DL' PAGE 'PNL' "	heading clause

The BREAK-ON option string is as follows:

BREAK-ON STATE "TOT 'LUBP'"

The text string "TOT" is printed at the break line. The break options, 'LUBP', suppress the break skip line, underline the TOTALS, place the break value in the HEADING, and force a new page on each break, respectively.

The value of the break referenced by the option 'B' must have a "sister" option 'B' in the HEADING statement.

HEADING "BALANCE FOR 'B' ON 'DL' PAGE 'PNL' "

The B option accepts the break value in effect when the new page is generated. The remaining options display the date, perform a line feed, generate the current page number and perform another line feed. The output for the first page is as follows:

```
BALANCE FOR CA ON 06 DEC 1988  
PAGE 1  
  
CUSTOMERS. STATE COMPANY..... Balance Due.  
  
102      CA      THE GUN STORE              12.34  
103      CA      AEROBIC DEN                89.12  
105      CA      DIABOLICAL SOFTWARE         4.56  
106      CA      PHINK, PHINK AND PHI        126.75  
          NK  
107      CA      MONA LISA LTD              789.32  
  
          TOT                                -----  
                                     1022.09
```

Press <return> to see each subsequent page, or enter a <ctl>X (hold the control key, <ctl> and the X key simultaneously) to abort and return to TCL.

4.7

OTHER OUTPUT MODIFIERS

Output modifiers are used to change the default output characteristics of ACCESS reports.

NOTE

Each modifier can be replaced with a corresponding TCL option surrounded by parenthesis, (option). The list of the additional output modifiers is as follows:

MODIFIER	OPTION	DESCRIPTION
ID-SUPP	(I)	Suppresses the automatic output of the item-id column on the report.
DET-SUPP	(D)	Used in an ACCESS statement in which a sort sequence is combined with a BREAK-ON modifier. This suppresses the output of all the detail lines between each break line and, instead, displays the data value that caused the break to occur.
HDR-SUPP	(H)	Suppresses the default page heading that contains the time, date, and page number. Please do not use in conjunction with the HEADING or FOOTING modifiers. That would be self defeating, to say the least.
COL-HDR-SUPP	(C)	Suppresses both the default page heading which normally includes time, date, and page number, as well as the data column headings.
NOPAGE	(N)	Causes the report to output to the terminal display (default) without pausing for the user to enter a <carriage return> between each page. This has no effect if the output is directed to the line printer via the LPTR modifier or (P) option. NOPAGE is typically used when the report is printed on a terminal attached (slave) printer.
LPTR	(P)	Direct output to the system printer via the spooler. For more information see Section 4.8, "Choosing the Output Device."

Here are samples of these modifiers. Enter the following sentence:

```
>SORT CUSTOMERS BY STATE BREAK-ON STATE COMPANY TOTAL BALANCE ID-SUPP<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
BREAK-ON STATE COMPANY...	
...TOTAL BALANCE	output-list
ID-SUPP	output modifier

>SORT CUSTOMERS BY STATE BREAK-ON STATE COMPANY TOTAL BALANCE ID-SUPP<r>

PAGE 1

12:31:26 06 DEC 1988

STATE. COMPANY..... BALANCE DUE.

CA	THE GUN STORE	12.34
CA	AEROBIC DEN	89.12
CA	DIABOLICAL SOFTWARE	4.56
CA	PHINK, PHINK AND PHI	126.75
	NK	
CA	MONA LISA LTD	789.32
***		1022.09
NY	SHIPS R US	567.80
NY	ACME CARROT	56.75
***		624.55
PA	SPEAK EASY	23.42
***		23.42
		1670.06

8 ITEMS LISTED.

The item-id column is not reported.

Now, enter the next sentence:

>SORT CUSTOMERS BY STATE BREAK-ON STATE TOTAL BALANCE DET-SUPP<r>

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
BREAK-ON STATE...	
...TOTAL BALANCE	output-list
DET-SUPP	output modifier

>SORT CUSTOMERS BY STATE BREAK-ON STATE TOTAL BALANCE DET-SUPP<r>

PAGE 1

12:31:39 06 DEC 1988

CUSTOMERS. STATE Balance Due.

CA	1022.09
NY	624.55
PA	23.42
***	1670.06

8 ITEMS LISTED.

Only the break lines are reported. Notice that the value of the break is automatically reported at the break line without having to use the 'V' BREAK-ON option.

Try the rest of these modifiers on your own.

4.8

CHOOSING THE OUTPUT DEVICE

The directed output device can be a terminal or a printer, depending on whether the LPTR (or P option) modifier is or is not used in the ACCESS sentence.

```
>LIST CUSTOMERS COMPANY CONTACT<r>
```

sends output to the terminal screen.

```
>LIST CUSTOMERS COMPANY CONTACT LPTR<r>
```

or

```
>LIST CUSTOMERS COMPANY CONTACT (P)<r>
```

sends output to the system printer.

In the examples in this chapter so far, the sum of the width of the columns does not exceed the width of the current output device. If, however, the sum of the column width exceeds the width of the output device, the output is listed vertically along the side of the page.

For example:

```
>LIST CUSTOMERS COMPANY CONTACT ADDRESS CITY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
COMPANY CONTACT...	
...ADDRESS CITY STATE	output-list

generates the following output to the terminal.

```
>LIST CUSTOMERS COMPANY CONTACT ADDRESS CITY STATE<r>
```

```
PAGE 1
```

```
13:33:40 08 SEP 1988
```

```
CUSTOMERS : 106
```

```
COMPANY PHINK, PHINK AND PHINK
```

```
CONTACT PHINK, CLYDE
```

```
ADDRESS 1600 PENNSYLVANIA AVE
```

```
CITY SANTA ANA
```

```
STATE CA
```

```
CUSTOMERS : 107
```

```
COMPANY MONA LISA LTD
```

```
CONTACT DAVINCI, LEONARDO
```

```
ADDRESS 15 CENTURY DR
```

```
CITY VENICE BEACH
```

```
STATE CA
```

CUSTOMERS : 100
COMPANY SHIPS R US
CONTACT HORNBLOWER, HORATIO
ADDRESS 1110-1 W 57TH ST
CITY NEW YORK
STATE NY

CUSTOMERS : 101
COMPANY ACME CARROT
CONTACT BUNNY, BUGS
ADDRESS 10 HOLE-IN-GROUND
CITY BROOKLYN
STATE NY

CUSTOMERS : 102
COMPANY THE GUN STORE
CONTACT FUDD, ELMER
ADDRESS 30 WABBIT WAY
CITY WEDONDO BEACH
STATE CA

CUSTOMERS : 103
COMPANY AEROBIC DEN
CONTACT PORCELANA, PORCHE
ADDRESS 139 HOLLYWOOD BLVD.
CITY HOLLYWOOD
STATE CA

CUSTOMERS : 104
COMPANY SPEAK EASY
CONTACT PARTICIPLE, DANGLY
ADDRESS 3 DAT DARE ST
CITY GRAMMAR CITY
STATE PA

CUSTOMERS : 105
COMPANY DIABOLICAL SOFTWARE
CONTACT GROG, BILL SPIDER
ADDRESS 01100 BINARY WAY
CITY BYTELAND
STATE CA

8 ITEMS LISTED.

The total column length exceeds the width of the terminal.

The width and depth of the terminal or printer is controlled by the TCL command TERM. The TERM command allows you to temporarily change the width and length of the terminal or printer for the current logon session. The minute that you log off and log back on, the TERM characteristics are reset to the current system default values.

The TERM statement by itself displays the current device parameters.

>TERM<r>

TERMINAL PRINTER		
PAGE WIDTH:	79	132
PAGE DEPTH:	25	60
LINE SKIP :	0	
LF DELAY :	0	
FF DELAY :	2	
BACKSPACE :	8	
TERM TYPE :	I	

Here is what each of these displayed elements mean:

PAGE WIDTH	The number of columns across the page for the terminal or the system printer.
PAGE DEPTH	The number of lines per page on the terminal or printer.

Please note that the page width and depth are parameters used by the system to determine when a line feed and carriage return is to be generated at the end of the current line (width), and when a form feed is to be output at the end of the current page (depth). The printer must be set for the correct form width (CPI, characters per inch), and form length (inches per page), or the page alignment will be off. Please consult your printer manuals to set up your printer correctly.

LINE SKIP	Both LINE SKIP and LF DELAY (Line Feed Delay) can be set to zero. These parameters are leftovers from the early days of PICK and have no bearing on what we are doing here. Therefore, you can ignore them.
LF DELAY	
FF DELAY	
	Form Feed Delay is another appendage which can, for the most part, be overlooked. However, there is one feature which can be invoked by using it.
	<i>If the FF DELAY is 2 or more (as usual), the screen and/or printer are sent a form feed (go to the next page) between each printed page.</i>
	<i>If the FF DELAY is 1, only the printer is sent a form feed. The terminal scrolls between pages without clearing the screen.</i>
	<i>If the FF DELAY is 0, both the terminal and printer scroll between pages without generating a form feed.</i>
	"What does this have to do with ACCESS and this discussion?"
	NOT MUCH! Try these parameters if you want.
	Otherwise, you don't have to bother.
BACKSPACE	Do not touch. This determines what your system port will recognize as a backspace character. It is not needed for this discussion.
TERM TYPE	The code which represents the type of terminal that you are using. Usually, the following characters are used:
	A - ADDS terminal
	V - Viewpoint
	D - Dec VT101 or VT102
	I - IBM memory mapped monitor
	W - Wyse 50
	However, this should not make a difference when generating ACCESS reports.

To prevent confusion when specifying a change to the device parameters, each position is given an identifying letter as follows:

	TERMINAL PRINTER	
PAGE WIDTH:	a	g
PAGE DEPTH:	b	h
LINE SKIP :	c	
LF DELAY :	d	
FF DELAY :	e	
BACKSPACE :	f	
TERM TYPE :	t	

When changing the TERM parameters, the general form is:

```
>TERM a,b,c,d,e,f,g,h,t<r>
```

Here are some examples:

Change the terminal width to 132 columns and the terminal depth to 60 lines.

```
>TERM 132,60<r>
```

Now verify the change.

```
>TERM<r>
```

	TERMINAL PRINTER	
PAGE WIDTH:	132	132
PAGE DEPTH:	60	60
LINE SKIP :	0	
LF DELAY :	0	
FF DELAY :	2	
BACKSPACE :	8	
TERM TYPE :	I	

Notice that the rest of the parameters remain unchanged.

Now change the printer width to 80 and the depth to 55. Changing the width and depth of the printer is a bit more cumbersome. If the parameters preceding the first changed parameter are to remain unchanged, then commas (,) must be used in their place. The (R) option requests a redisplay of the changed parameters without having to re-enter the TERM command.

```
>TERM ,,,,,,80,55 (R)<r>
```

TERMINAL PRINTER

PAGE WIDTH:	132	80
PAGE DEPTH:	60	55
LINE SKIP :	0	
LF DELAY :	0	
FF DELAY :	2	
BACKSPACE :	8	
TERM TYPE :	I	

Changing the terminal type is a bit easier. This parameter can "stand alone."

```
>TERM V<r>
```

This terminal is now set to emulate a Viewpoint terminal.

EXERCISE**4**

1. The verb which generates a columnar report in random order is _____.
2. The verb which generates a columnar report in sorted order is _____.
3. How many columns of output does the following ACCESS sentence generate? _____

 >LIST CUSTOMERS WITH STATE = "CA" COMPANY CONTACT PHONE
4. Name each output column generated by the above statement.

5. _____ formatting allows columns to be automatically reported.
6. Data definition items with numeric item-ids are the default _____ in an implicit report.
7. The _____ modifier overrides implicit formatting so that only the item-id column is reported.
8. Match the following modifiers with the correct description:

_____ DET-SUPP	a. Generates a visual break line when the data definition value changes.
_____ FOOTING	b. Allows specification of a customized report heading.
_____ GRAND-TOTAL	c. Suppresses page headers.
_____ (P)	d. Suppresses column and page headers.
_____ HDR-SUPP	e. Suppresses the default item-id column.
_____ BREAK-ON	f. Allows specification of a customized report footing.
_____ (H)	g. The print option. The same as the LPTR output modifier.
_____ TOTAL	h. Suppresses detail lines on a report.
_____ HEADING	i. Generates columnar totals at the break lines and on the grand total line.
_____ ID-SUPP	j. Allows custom text on the grand total line.
_____ COL-HDR-SUPP	k. The option synonym for HDR-SUPP.

9. Enter the HEADING clause which says to: skip a line, center the word HEADER, and skip two lines.
-
10. The BREAK-ON option string to print the value of the break and underline the totals is:
-
11. Enter the ACCESS sentence for the CUSTOMERS file which sends to the system printer the total balance due for each state without the details.
-
12. Enter the statement to set the terminal width to 132 and depth to 65.
-
13. Enter the statement to set the line printer to a width of 80 and a depth of 55 with an automatic re-display of the parameters.
-
14. Match the following:
- | | |
|--|--|
| _____ GRAND-TOTAL "TOTAL BALANCE" | a. Generate a page number, centered at the bottom of each page. |
| _____ HEADING "A HEADING" | b. Display the break value at the break line and force a new page. |
| _____ BREAK-ON STATE TOTAL BALANCE | c. Print A HEADING at the beginning of every page. |
| _____ BREAK-ON STATE "'PV'" | d. Force a new page for each new state and place the break value in the heading. |
| _____ BREAK-ON STATE "'U'"
TOTAL BALANCE | e. Show the balance due for each state. |
| _____ FOOTING "'LC'PAGE 'PN'" | f. Display the text TOTAL BALANCE on the grand total line. |
| _____ BREAK-ON STATE "'PB'"
HEADING "FOR STATE 'B'" | g. Underline the total balance due for each state. |

5

MAILING LABELS

5.1

LABEL VERBS AND PARAMETERS

ACCESS can be used to generate mailing labels by using either of the following verbs.

VERB	DESCRIPTION
LIST-LABEL	Generates labels from a file in random order.
SORT-LABEL	Generates labels from a file in a specified sorted order.

Columnar ACCESS commands generate reports in the following format:

item 1	attr a	attr b	attr n
item 2	attr a	attr b	attr n
item 3	attr a	attr b	attr n

The LABEL verbs output data in the following format:

item 1 attr a attr b	item 2 attr a attr b	item 3 attr a attr b
item 4 attr a attr b	item 5 attr a attr b	item 6 attr a attr b

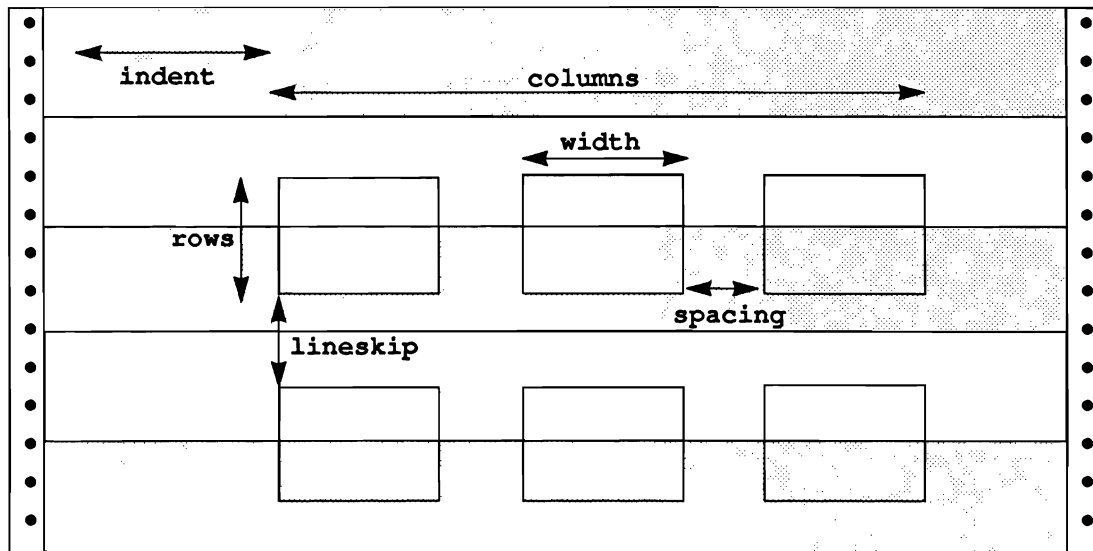
Whenever the LABEL verbs are used, the system prompts for a set of parameters which indicate the format of the labels on the page. THERE IS NO DEFAULT! After the command sentence is entered, the only prompt displayed on the screen is a question (?) mark. The label parameters must be specified in the following format:

?columns,rows,lineskip,indent,width,spacing,C<r>

These seven parameters represent the following:

PARAMETER	DESCRIPTION
columns	The number of labels in a set across the page.
rows	The number of lines per label.
lineskip	The number of blank lines between each set of labels. A vertical skip.
indent	The number of spaces indented from left margin to begin printing the labels.
width	The number of character positions wide for each label.
spacing	The number of spaces between each label in a set. Horizontal spacing.
C	The compress option. This compresses null values in a label so that no blank lines appear within the body of the label.

Here is a visual representation of the parameter meanings:



Enter the following command and provide the indicated parameter response:

```
>LIST-LABEL CUSTOMERS COMPANY CONTACT CITY STATE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST-LABEL	verb
CUSTOMERS	filename
COMPANY CONTACT CITY STATE	output-list

Now enter the following:

```
?2,4,1,0,30,1,C<r>
```

The parameters indicate:

- 2 labels across the page
- 4 lines per label
- 1 line vertically between labels
- 0 no indent, start at the left margin
- 30 characters per label line
- 1 space horizontally between the labels
- C compress the null values

The output is as follows:

```
>LIST-LABEL CUSTOMERS COMPANY CONTACT CITY STATE<r>
```

```
?2,4,1,0,30,1,C<r>
```

```
PAGE 1
```

```
14:38:22 07 DEC 1988
```

```
106
```

```
PHINK, PHINK AND PHINK  
PHINK, CLYDE  
SANTA ANA
```

```
107
```

```
MONA LISA LTD  
DAVINCI, LEONARDO  
VENICE BEACH
```

```
100
```

```
SHIPS R US  
HORNBLLOWER, HORATIO  
NEW YORK
```

```
101
```

```
ACME CARROT  
BUNNY, BUGS  
BROOKLYN
```

```
102
```

```
THE GUN STORE  
FUDD, ELMER  
WEDONDO BEACH
```

```
103
```

```
AEROBIC DEN  
PORCELANA, PORCHE  
HOLLYWOOD
```

```
104
```

```
SPEAK EASY  
PARTICIPLE, DANGLY  
GRAMMAR CITY
```

```
105
```

```
DIABOLICAL SOFTWARE  
GROG, BILL SPIDER  
BYTELAND
```

Notice that the STATE is missing. This is because only four lines are allowed per label, but the total number of output lines is five. Yes, five. The item-id is automatically output, just like a columnar report. See section 5.2, "Using Output Modifiers," to learn how to get around this.

Possible Errors:

The label parameters used must be precisely balanced or the LIST-LABEL statement will fail. This means that the total width of the output cannot exceed the width of the page. For example, re-entered using the following parameters:

```
?3,4,1,0,30,1,C<r>
```

In this case, we are asking for three labels across the page with 30 columns per label. Well, 3 times 30 is 90, which is far wider than the current 79 page width. The following error message is displayed:

[290] THE RANGE OF THE PARAMETER "1" IS NOT ACCEPTABLE.

The parameter indicated in the error message is the number of horizontal spaces between labels. However, less than one space between labels is hard if not impossible to read. To fix this, the label width should be reduced. 20 is a good width since 20 times 3 is 60.

?3,4,1,0,20,1,C<r>

This produces:

>LIST-LABEL CUSTOMERS COMPANY CONTACT CITY STATE<r>

?3,4,1,0,20,1,C<r>

PAGE 1

12:14:49 20 DEC 1988

Truncates output to 20 characters.

106 PHINK, PHINK AND PHI PHINK, CLYDE SANTA ANA	107 MONA LISA LTD DAVINCI, LEONARDO VENICE BEACH	100 SHIPS R US HORNBLOWER, HORATIO NEW YORK
101 ACME CARROT BUNNY, BUGS BROOKLYN	102 THE GUN STORE FUDD, ELMER WEDONDO BEACH	103 AEROBIC DEN PORCELANA, PORCHE HOLLYWOOD
104 SPEAK EASY PARTICIPLE, DANGLY GRAMMAR CITY	105 DIABOLICAL SOFTWARE GROG, BILL SPIDER BYTELAND	

5.2

USING OUTPUT MODIFIERS

In the above example, there is more than just the item-id which needs to be suppressed. The output also has a heading, which is usually not desirable when printing mailing labels. The following output modifiers are the most frequently used when generating labels.

MODIFIER	OPTION	DESCRIPTION
ID-SUPP	(I)	Suppresses the automatic output of the item-id line on the label.
HDR-SUPP	(H)	Suppresses both the default page heading which normally includes time, date, and page number, as well as the label line headers.
LPTR	(P)	Send output to the system printer via the spooler.

It is common practice to use both ID-SUPP and HDR-SUPP when generating mailing labels.

Enter the next command sentence along with the indicated parameter response:

>LIST-LABEL CUSTOMERS COMPANY CONTACT CITY STATE ID-SUPP HDR-SUPP<r>

? , 4 , 1 , 0 , 30 , 1 , C <r>

SENTENCE ELEMENT	DESCRIPTION
LIST-LABEL	verb
CUSTOMERS	filename
COMPANY CONTACT CITY STATE	output-list
ID-SUPP HDR-SUPP	output modifiers

>LIST-LABEL CUSTOMERS COMPANY CONTACT CITY STATE ID-SUPP HDR-SUPP<r>

? 2 , 4 , 1 , 0 , 30 , 1 , C <r>

PHINK, PHINK AND PHINK	MONA LISA LTD
PHINK, CLYDE	DAVINCI, LEONARDO
SANTA ANA	VENICE BEACH
CA	CA

SHIPS R US	ACME CARROT
HORNBLOWER, HORATIO	BUNNY, BUGS
NEW YORK	BROOKLYN
NY	NY

THE GUN STORE	AEROBIC DEN
FUDD, ELMER	PORCELANA, PORCHE
WEDONDO BEACH	HOLLYWOOD
CA	CA

SPEAK EASY	DIABOLICAL SOFTWARE
PARTICIPLE, DANGLY	GROG, BILL SPIDER
GRAMMAR CITY	BYTELAND
PA	CA

Notice that suppressing the item-id allows all of the requested fields to be displayed. In addition, the output has no page header.

This same command can be used to print the labels on the system printer by just including the LPTR output modifier. Enter the following command sentence which indicates ID-SUPP, HDR-SUPP, and LPTR, using the equivalent output options.

>LIST-LABEL CUSTOMERS COMPANY CONTACT CITY STATE (IHP)<r>

SENTENCE ELEMENT	DESCRIPTION
LIST-LABEL	verb
CUSTOMERS	filename
COMPANY CONTACT CITY STATE	output-list
(IHP)	output options

The output is now directed to the system printer.

5.3

USING INDENT

A non-zero indent parameter has a tendency to alter the flow of the parameter specification process. When the value of indent is not zero, the LIST-LABEL process assumes that each row is to have a descriptive heading as follows:

•					•
•					•
•	rowhead	item 1	item 2	item 3	•
•	rowhead	attr a	attr a	attr a	•
•	rowhead	attr b	attr b	attr b	•
•					•
•					•
•					•

Therefore, after accepting the initial parameter string, the LIST-LABEL requests entry of a header for each row. In the next example, the indent is set to ten, and the number of rows is four. This causes four more question (?) marks prompting for operator response.

Enter the following command sentence:

```
>LIST-LABEL CUSTOMERS COMPANY CONTACT CITY STATE (HI)<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST-LABEL	verb
CUSTOMERS	filename
COMPANY CONTACT CITY STATE	output-list
(HI)	options

```
?2, 4, 1, 10, 30, 1, C<r>
?Company<r>
?Contact<r>
?City<r>
?State<r>
```

The row heading text is entered at each subsequent prompt. A <return> key should be entered for each of the row headings if no text is required. The output is:

>LIST-LABEL CUSTOMERS COMPANY CONTACT CITY STATE (HI)<r>

?2,4,1,10,30,1,C<r>

?Company<r>

?Contact<r>

?City<r>

?State<r>

Company PHINK, PHINK AND PHINK
Contact PHINK, CLYDE
City SANTA ANA
State CA

MONA LISA LTD
DAVINCI, LEONARDO
VENICE BEACH
CA

Company SHIPS R US
Contact HORNBLLOWER, HORATIO
City NEW YORK
State NY

ACME CARROT
BUNNY, BUGS
BROOKLYN
NY

Company THE GUN STORE
Contact FUDD, ELMER
City WEDONDO BEACH
State CA

AEROBIC DEN
PORCELANA, PORCHE
HOLLYWOOD
CA

Company SPEAK EASY
Contact PARTICIPLE, DANGLY
City GRAMMAR CITY
State PA

DIABOLICAL SOFTWARE
GROG, BILL SPIDER
BYTELAND
CA

NOTE

On some implementations, the row header defaults to the standard column header used in a columnar report. To eliminate this, the sentence must be modified with the COL-HDR-SUPP output modifier. Please consult your manuals for details.

EXERCISE

5

1. The two ACCESS verbs used for generating mailing labels are:

2. Identify each element of the label parameter string:
?a,b,c,d,e,f,g

a. _____

b. _____

c. _____

d. _____

e. _____

f. _____

g. _____
3. Write the ACCESS sentence and provide the parameters to generate mailing labels from the CUSTOMERS file using the following requirements:
 - The labels should contain company name, address, city, state and zip on separate lines.
 - The item-id, and all headings, should be suppressed.
 - The output should be sent to the printer.
 - Three labels across an 80 column page.
> _____<r>

? _____<r>

6

ACCESS LISTS

ACCESS can be thought of as processing in three main phases. The first parses (takes apart) and compiles the sentence so that the system understands what you want. The second performs the data sampling, selection and sequencing. The resulting list of item-ids is used for the third phase which generates the output. ACCESS can be intercepted at the completion of the second phase. In this way, ACCESS is used as a pre-processor, effectively "weeding out" unwanted data before passing the "good" items to any other TCL, ACCESS or PICK/BASIC routines for further processing.

6.1 SELECTING AND USING LISTS

Lists are the result of the SELECT and SSELECT verbs.

SELECT	Generates a list of item identifiers in random order which meet the criteria of the selection clause. This list is a subset of the total file and may be saved and used later for other processing.
SSELECT	Functionally identical to SELECT but provides for the sequencing of item-ids by one or more sort criteria.

The general form of the SELECT verbs is as follows:

```
>SELECT filename {selection clause}<r>
```

```
>SSELECT filename {sequence clause} {selection clause}<r>
```

Enter this command sentence:

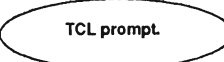
```
>SELECT CUSTOMERS WITH STATE = "CA"<r>
```

SENTENCE ELEMENT	DESCRIPTION
SELECT	verb
CUSTOMERS	filename
WITH STATE = "CA"	selection clause

The system responds with:

```
>SELECT CUSTOMERS WITH STATE = "CA"<r>
```

```
[404] 5 ITEMS SELECTED.
```

> 

A SELECT or SSELECT generates what is called an active list. At this point, the system expects the immediate entry of a command sentence which invokes a process using the currently active list.

At the TCL (>) prompt, enter the following:

```
>LIST CUSTOMERS COMPANY STATE<r>
```

The output is:

```
>LIST CUSTOMERS COMPANY STATE<r>
PAGE 1                      12:23:40  08 DEC 1988

CUSTOMERS. COMPANY..... STATE
106          PHINK, PHINK AND PHI CA
              NK
107          MONA LISA LTD          CA
102          THE GUN STORE          CA
103          AEROBIC DEN            CA
105          DIABOLICAL SOFTWARE    CA

5 ITEMS LISTED.
```

Only those items selected previously are reported by the subsequent LIST command. So, using a SELECT is a two step process. First, the active list is built via the SELECT command sentence, and second, the list is used by the next command entered at TCL.

Once processed, the list is no longer active. Any command entered immediately after the select (even if it doesn't use the list), deactivates the list.

Try this again. Enter the following:

```
>SSELECT CUSTOMERS BY STATE BY COMPANY<r>
```

SENTENCE ELEMENT	DESCRIPTION
SSELECT	verb
CUSTOMERS	filename
BY STATE BY COMPANY	sequence clause

The system responds with:

```
>SSELECT CUSTOMERS BY STATE BY COMPANY<r>
```

```
[404] 8 ITEMS SELECTED.
```

```
>
```

TCL prompt.
The list is active.

Now immediately enter:

```
>LIST CUSTOMERS BREAK-ON STATE COMPANY<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
BREAK-ON STATE COMPANY	output-list

The result is:

```
>LIST CUSTOMERS BREAK-ON STATE COMPANY<r>

PAGE 1                                12:27:47  08 DEC 1988

CUSTOMERS. STATE COMPANY.....

103      CA      AEROBIC DEN
105      CA      DIABOLICAL SOFTWARE
107      CA      MONA LISA LTD
106      CA      PHINK, PHINK AND PHI
              NK
102      CA      THE GUN STORE

              ***

101      NY      ACME CARROT
100      NY      SHIPS R US

              ***

104      PA      SPEAK EASY

              ***

***

8 ITEMS LISTED.
```

Notice that both of the above examples display the data selected and sequenced by the previous SELECT and SSELECT commands.

As stated above, an active list can also be passed to another TCL process other than ACCESS. Try the following:

```
>SSELECT CUSTOMERS BY COMPANY WITH STATE = "CA"<r>

[404] 5 ITEMS SELECTED.
```

>

The list is active.

Now immediately enter the following:

```
>ED CUSTOMERS<r>
```

Notice that no item-list is required when an active list is being passed to the command sentence.

```
>ED CUSTOMERS<r>
103
TOP
.L5<r>
001 AEROBIC DEN
002 PORCELANA, PORCHE
003 139 HOLLYWOOD BLVD
004 HOLLYWOOD
005 CA
.EX<r>
'103' EXITED
105
TOP
.L5<r>
001 DIABOLICAL SOFTWARE
002 GROG, BILL SPIDER
003 01100 BINARY WAY
004 BYTELAND
005 CA
.EX<r>
'105' EXITED
107
TOP
.L5<r>
001 MONA LISA LTD
002 DAVINCI, LEONARDO
003 15 CENTURY DR
004 VENICE BEACH
005 CA
.EXK<r>
'107' EXITED
```

Callouts:

- Edit the items on the active list in CUSTOMERS.
- List 5 lines.
- Exit to the next item in the list.
- All items have "CA" in attribute 005.
- Exit to TCL and kill the list.

>

The Editor only acts upon those items in the active list in the sequence that were selected. Re-enter the previous SELECT sentence and enter the following TCL sentence immediately afterwards.

```
>SSELECT CUSTOMERS BY COMPANY WITH STATE = "CA"<r>
```

SORTs and SELECTs all customers in California and orders the list by company name.

```
>COPY CUSTOMERS (T)<r>
```

Copies to the terminal all items identified in the active list.

This says to copy the item image of all of the CUSTOMERS items in the state of California to the terminal. Observe what happens.

CATALOGUED LISTS

A list which is active only until the next command is invoked requires that the file be re-SELECTed every time a new process needs the same list. This can get quite time consuming on a large file. However, this need not be the case. A list can be built and then catalogued so that it may be retrieved numerous times.

Catalogued (or saved) lists must be stored in a file called POINTER-FILE which resides in each account. Before continuing, make sure that your account has a POINTER-FILE by using the following procedure:

Enter:

```
>LIST POINTER-FILE<r>
```

If POINTER-FILE does not exist, the system will display the following message:

```
[201] 'POINTER-FILE' IS NOT A FILENAME
```

To create a POINTER-FILE, enter these commands:

```
>CREATE-PFILE DICT POINTER-FILE 29,1<r>
```

```
[417] FILE 'POINTER-FILE' CREATED; BASE = 10082, MODULO = 29, SEPAR = 1.
```

This shows how to create a pointer type file with a modulo of 29 and a separation of 1. If your implementation does not have the CREATE-PFILE command, then enter the following:

```
>CREATE-FILE DICT POINTER-FILE 29,1<r>
```

Now you must identify this file as a pointer type file. Enter the following:

```
>ED MD POINTER-FILE<r>
TOP
.1<r>
001 D
.R/D/DC<r>
001 DC
.FI<r>
'POINTER-FILE' FILED
```

Edit the pointer in the master dictionary.

Goto attribute 001.

Replace the D in attribute 001 with a DC indicating a pointer type file.

File and exit the D pointer.

Don't worry about the nuances of a pointer class file versus a regular file. You only have to worry about doing what has been just described if there isn't a pointer-file already defined on your account.

6.2.1

Saving Lists

A list is catalogued by entering the following verb immediately after performing a SELECT.

VERB	DESCRIPTION
SAVE-LIST	Saves and catalogs an active list to the POINTER-FILE. After invocation of SAVE-LIST, the list is no longer active.

The general form is:

```
SAVE-LIST listname<r>
```

The "listname" is any valid name (item-id) without imbedded blanks. SAVE-LIST must be performed when a list is active, otherwise there is nothing to save. Enter the following:

```
>SSELECT CUSTOMERS BY STATE BY COMPANY<r>
```

```
[404] 8 ITEMS SELECTED.
```

System message.

```
>SAVE-LIST CUSTS<r>
```

```
List 'CUSTS' saved. 1 frame/s used.
```

```
>
```

TCL prompt. The list is no longer active.

The active list is now saved for later retrieval under the item-id CUSTS. When a list is saved to the POINTER-FILE, a special item is written which points to the location of the list on the disk. The message displayed after a SAVE-LIST indicates how much disk space the list uses in frames. A frame is a measurement of disk space...well, don't worry about it!

6.2.2

Retrieving Lists

VERB	DESCRIPTION
GET-LIST	Retrieves and activates a list stored in POINTER-FILE.

The general form is:

```
>GET-LIST listname<r>
```

GET-LIST can be used just like a SELECT. The active list can now be passed to the command invoked immediately following the GET-LIST.

Enter the following:

```
>GET-LIST CUSTS<r>
```

```
[404] 8 ITEMS SELECTED.
```

System message.

Now enter:

```
>LIST CUSTOMERS BREAK-ON STATE COMPANY<r>
```

The report output is:

```
>GET-LIST CUSTS<r>
```

```
[404] 8 ITEMS SELECTED.
```

System message.

```
>LIST CUSTOMERS BREAK-ON STATE COMPANY<r>
```

```
PAGE 1
```

```
14:30:01 08 DEC 1988
```

```
CUSTOMERS. STATE COMPANY.....
```

```
103      CA      AEROBIC DEN
105      CA      DIABOLICAL SOFTWARE
107      CA      MONA LISA LTD
106      CA      PHINK, PHINK AND PHI
              NK
102      CA      THE GUN STORE
```

```
***
```

```
101      NY      ACME CARROT
100      NY      SHIPS R US
```

```
***
```

```
104      PA      SPEAK EASY
```

```
***
```

```
***
```

```
8 ITEMS LISTED.
```

The BREAK-ON modifier works correctly with the LIST command because the 'CUSTS' list is already in sorted order by state. Try this next command sequence on your own:

```
>GET-LIST CUSTS<r>
```

```
[404] 8 ITEMS SELECTED.
```

```
>COPY CUSTS (T)<r>
```

6.3

REPORTING CATALOGUED LISTS

Since catalogued lists are represented by items in a file, POINTER-FILE to be exact, they can be reported using an ACCESS command sentence.

To simply see what lists already exist, enter the following:

```
>SORT ONLY POINTER-FILE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
ONLY	modifier
POINTER-FILE	filename

The output may look like:

```
>SORT ONLY POINTER-FILE<r>
```

```
PAGE 1
```

```
11:43:39 12 DEC 1988
```

```
POINTER-FILE
```

```
XX
```

```
POINTER-FILE
```

```
00.FILES
```

```
HAN
```

```
WORDDUMPLIST
```

```
CUSTS
```

```
6 ITEMS LISTED.
```

There are six lists currently saved on my test account. Your results may vary.

This is all well and good, but it doesn't tell much about the state of the list. Sometimes, what is needed is information like what time and date the list was catalogued, and how big the list is. Enter the following:

```
>LIST-ITEM DICT POINTER-FILE 'CUSTS'<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST-ITEM DICT	verb
POINTER-FILE	filename
'CUSTS'	item-list

Here is what the output looks like:

```
>LIST-ITEM DICT POINTER-FILE 'CUSTS'<r>

PAGE 1                                14:12:43  08 DEC 1988

      CUSTS
001 CL
002 8893
003 1
004 8
005 14:06:58  08 DEC 1988

[404] 1 ITEMS LISTED.
```

The layout of the above POINTER-FILE item is as follows:

ITEM	DESCRIPTION
001 CL	Always a CL for a Catalogued List.
002 Disk location	The frame number on disk where the list begins.
003 Size	The number of frames used.
004 Items	The number of entries on the list.
005 Time/Date	The time and date that the list was catalogued.

The verbs LIST-ITEM or SORT-ITEM output the entire item in logical format. Logical item format is discussed in Chapter 1, "System Overview."

To get all the list pointers, enter the following:

```
>SORT-ITEM DICT POINTER-FILE WITH *A1 = "CL"<r>

SENTENCE ELEMENT  DESCRIPTION
SORT-ITEM         verb
DICT POINTER-FILE filename
WITH *A1 = "CL"   selection clause
```

The selection clause WITH *A1 = "CL" uses a definition name of *A1. Where did *A1 come from? *A1 is found in the master dictionary, MD. There are a number of default definitions in MD which can be used to display any attribute of an item.

These display attributes 1 through 5.

*A1 *A2 *A3 *A4 *A5

Definition items are looked up in MD under two circumstances:

- The ACCESS sentence is referencing the file DICT.
- or
- The definition item cannot be found in the file DICT.

The literal "CL" is used because all list pointers in POINTER-FILE happen to have a CL in the first attribute.

A report of the POINTER-FILE can also be generated by the following:

```
>SORT DICT POINTER-FILE *A1 *A2 *A3 *A4 *A5<r>
```

6.4 **LIST MAINTENANCE**

List maintenance consists of being able to:

- Copy a list to another name for backup or modification.
- Delete a list no longer in use.
- Revise (Edit) a list to add or remove entries manually.

6.4.1 ***Copying Lists***

VERB	DESCRIPTION
COPY-LIST	Copies a specified list to a new list name.

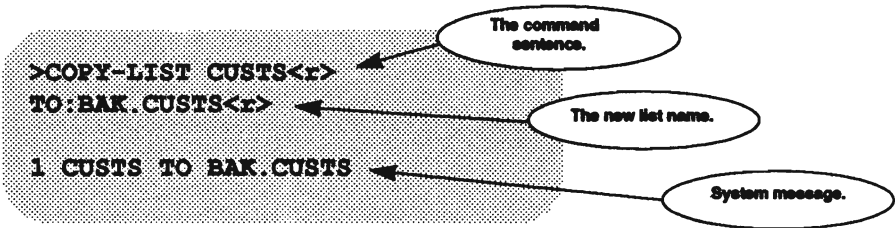
COPY-LIST is useful for backing up the current version of a list before using EDIT-LIST to alter the contents. The COPY-LIST verb follows an identical interaction to the COPY verb, except that the filename is defaulted to POINTER-FILE. Lists can be copied to new list names within the POINTER-FILE or copied as items to other PICK files.

The general form to copy a list to a new list name is:

```
>COPY-LIST listname<r>
TO:newlistname<r>
```

The TO: prompt is displayed by the system as a response to the COPY-LIST command sentence. The new list name must not already exist since, like COPY, COPY-LIST does not overwrite unless the (O) option is used. See the section on the options a little further on in this section.

Try this:



Now, use LIST-ITEM for the POINTER-FILE as described in section 6.3, "Reporting Catalogued Lists" and notice the components of the new list item.

```
>LIST-ITEM POINTER-FILE 'BAK.CUSTS'<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST-ITEM	verb
POINTER-FILE	filename
'BAK.CUSTS'	item-list

```
>LIST-ITEM POINTER-FILE 'BAK.CUSTS'<r>
```

```
PAGE 1
```

```
11:20:11 12 DEC 1988
```

```
BAK.CUSTS
```

```
001 CL
```

```
002 10177
```

```
003 1
```

```
004 8
```

```
005 10:55:27 12 DEC 1988
```

Sometimes, lists must be moved from the current POINTER-FILE into another file. The general form is:

```
>COPY-LIST listname<r>  
TO: (filename {newlistname})<r>
```

Notice that the new list name is in braces {}, indicating an optional specification. If a new list name is not indicated, then the original list retains its name in the new file.

The filename follows the same rules as do filename specifications throughout PICK. It must be preceded by a left parenthesis. To indicate the data level of CUSTS, the TO line reads:

```
TO: (CUSTOMERS
```

To indicate the dictionary level of the file, the TO line is specified:

```
TO: (DICT CUSTOMERS
```

Copy the CUSTS list to the dictionary of CUSTS as follows:

```
>COPY-LIST CUSTS<r>
```

```
TO: (DICT CUSTOMERS<r>
```

```
1 CUSTS TO CUSTS
```

The command sentence.

Copy to the dictionary of CUSTS, retaining the same list name.

System message.

Now, enter the following:

```
>LIST-ITEM DICT CUSTOMERS 'CUSTS'<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST-ITEM	verb
DICT CUSTOMERS	filename
'CUSTS'	item-list

Which generates the output:

```
>LIST-ITEM DICT CUSTOMERS 'CUSTS'<cr>
```

```
PAGE 1
```

```
11:16:16 12 DEC 1988
```

```
CUSTS
```

```
001 103
```

```
002 105
```

```
003 107
```

```
004 106
```

```
005 102
```

```
006 101
```

```
007 100
```

```
008 104
```

Notice that the list-type item has been converted to a regular old PICK item, no pointers, no save-date.

NOTE

Some newer PICK implementations no longer treat lists differently from items. Therefore, the list-item formats may differ. Please consult your manuals for any discrepancies that you may discover.

More than a single list can be copied at a time. The listname can be a series of listnames separated by spaces:

```
>COPY-LIST lista listb listc
```

or the listname can be an asterisk (*), indicating all items on file.

```
>COPY-LIST *
```

COPY-LIST Options:

These are the options which can be used to modify the COPY-LIST verb:

OPTION	DESCRIPTION
D	Delete the original item after copying.
O	Overwrite the new item if it already exists.
P	Copy the list contents to the system printer.
T	Copy the list contents to the terminal display.
N	For use with the T option only. Activates the NOPAGE mode.

Options are specified in the command sentence as all options on the system:

```
>COPY-LIST listname (options
```

The closed parenthesis is optional.

Try to copy BAK.CUSTS to the original CUSTS list as follows:

```
>COPY-LIST BAK.CUSTS<r>  
TO:CUSTS<r>
```

```
[415] 'BAK.CUSTS' EXISTS ON FILE.
```

System message.

The overwrite (O) option must be used to copy to an existing list. While you're at it, delete the BAK.CUSTS list. The command sentence is as follows:

```
>COPY-LIST BAK.CUSTS (O,D<r>  
TO:CUSTS<r>
```

```
1 BAK.CUSTS TO CUSTS
```

System message.

By the way, the commas are optional. Now, re-enter the following ACCESS sentence to confirm that BAK.CUSTS no longer exists.

```
>LIST ONLY POINTER-FILE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
ONLY	modifier
POINTER-FILE	filename

Using the P, T and N options can be done by entering each of the following commands:

```
>COPY-LIST CUSTS (P<r>
```

```
>COPY-LIST CUSTS (T<r>
```

and finally:

```
>COPY-LIST CUSTS (T,N<r>
```

6.4.2

Deleting Lists

VERB	DESCRIPTION
DELETE-LIST	Deletes a specified list-item.

The general form is:

```
>DELETE-LIST listname<r>
```

To demonstrate DELETE-LIST, first copy CUSTS back over to BAK.CUSTS as follows:

```
>COPY-LIST CUSTS<r>
TO:BAK.CUSTS<r>
```

Now list the POINTER-FILE to verify the BAK.CUSTS exists:

```
>SORT ONLY POINTER-FILE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
ONLY	modifier
POINTER-FILE	filename

Finally, delete BAK.CUSTS using the DELETE-LIST command:

```
>DELETE-LIST BAK.CUSTS<r>
```

```
[245] LIST 'BAK.CUSTS' DELETED!
```

System message.

Verify the removal of BAK.CUSTS with:

```
>SORT ONLY POINTER-FILE<r>
```

6.4.3

Editing Lists

VERB	DESCRIPTION
EDIT-LIST	Allows editing (alteration) of a LIST. Uses the text editor (EDIT) commands.

The general form of the command sentence is:

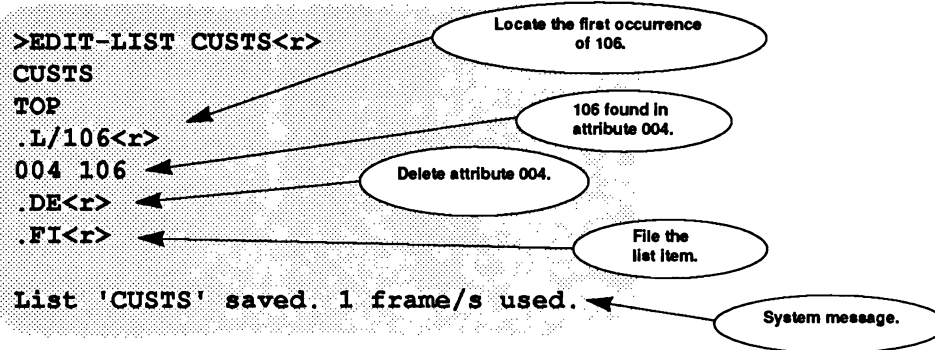
```
>EDIT-LIST listname<r>
```

The number of commands and the variations thereof make SP-EDIT difficult to address in this book. You must be familiar with Editor in order to use this verb. Ah hah! A good reason to review "HOW TO: The Editor," by yours truly. Anyway, here is a general list of the commands to use.

EDITOR COMMAND	DESCRIPTION
DEn	Delete n lines from the current line.
EX	Exit without filing.
FI	File the list and exit.
FD	Delete the list. Like DELETE-LIST.
Gn	Goto a specific line number.
I	Insert a line.
Ln	List n lines. Usually use L22 to list a full page.
L/string	Locate the first occurrence of string in the list item.
R	Replace an entire time.

To "limber up," try this next sample:

The goal is to eliminate from the report a single unwanted item without re-selecting the whole file. It just so happens that PHINK, PHINK, and PHINK, customer 106, has become a real fink and must be removed. Enter the following:



The list is re-catalogued when FILEd.

Now, when the GET-LIST is performed:

```
>GET-LIST CUSTS<r>

7 ITEMS SELECTED.
```

Magic! There is one less item in the active list. Now enter:

```
>LIST STAFF BREAK-ON STATE COMPANY<r>
```

EXERCISE

6

1. ACCESS has three major phases. The first phase performs _____, the second phase performs _____, and the third phase performs _____.
2. The two verbs which perform the first two phases alone are _____ and _____.
3. True or False. SELECT statements need an output-list. _____
4. True or False. An active list can be passed to a TCL, PICK/BASIC or another ACCESS process. _____
5. Write the ACCESS sentence for the CUSTOMERS file to select customers in the state of California in the order of the company name.

6. Match the following:

_____ SAVE-LIST S NAMES	a. Generate an active list of customer contacts whose last name begins with the letter S.
_____ SSELECT CUSTOMERS BY BALANCE	b. Activate the saved list S NAMES.
_____ SELECT CUSTOMERS WITH CONTACT = "S]"	c. Generate an active list of customers in order of the balance due.
_____ GET-LIST S NAMES	d. Save the currently active list and name it S NAMES.
_____ DELETE-LIST S NAMES	e. Delete the saved list S NAMES.
7. Fill in the blanks to create an active list of all the customers in California sorted by the balance due and save the list under the name CALCUSTS.
 > _____ CUSTOMERS _____ <r>
 [404] 5 ITEMS SELECTED
 > _____ CALCUSTS<r>

8. Fill in the blanks to reactivate CALCUSTS and list resulting customer's company name address and state.

> _____ CALCUSTS<r>

[404] 5 ITEMS SELECTED.

> _____<r>

9. Fill in the blanks to copy CALCUSTS to BAK.CALCUSTS in the POINTER-FILE.

> _____ CALCUSTS<r>

TO: _____<r>

10. Fill in the blanks to copy CALCUSTS to a file called LISTS.

> _____ CALCUSTS<r>

TO: _____<r>

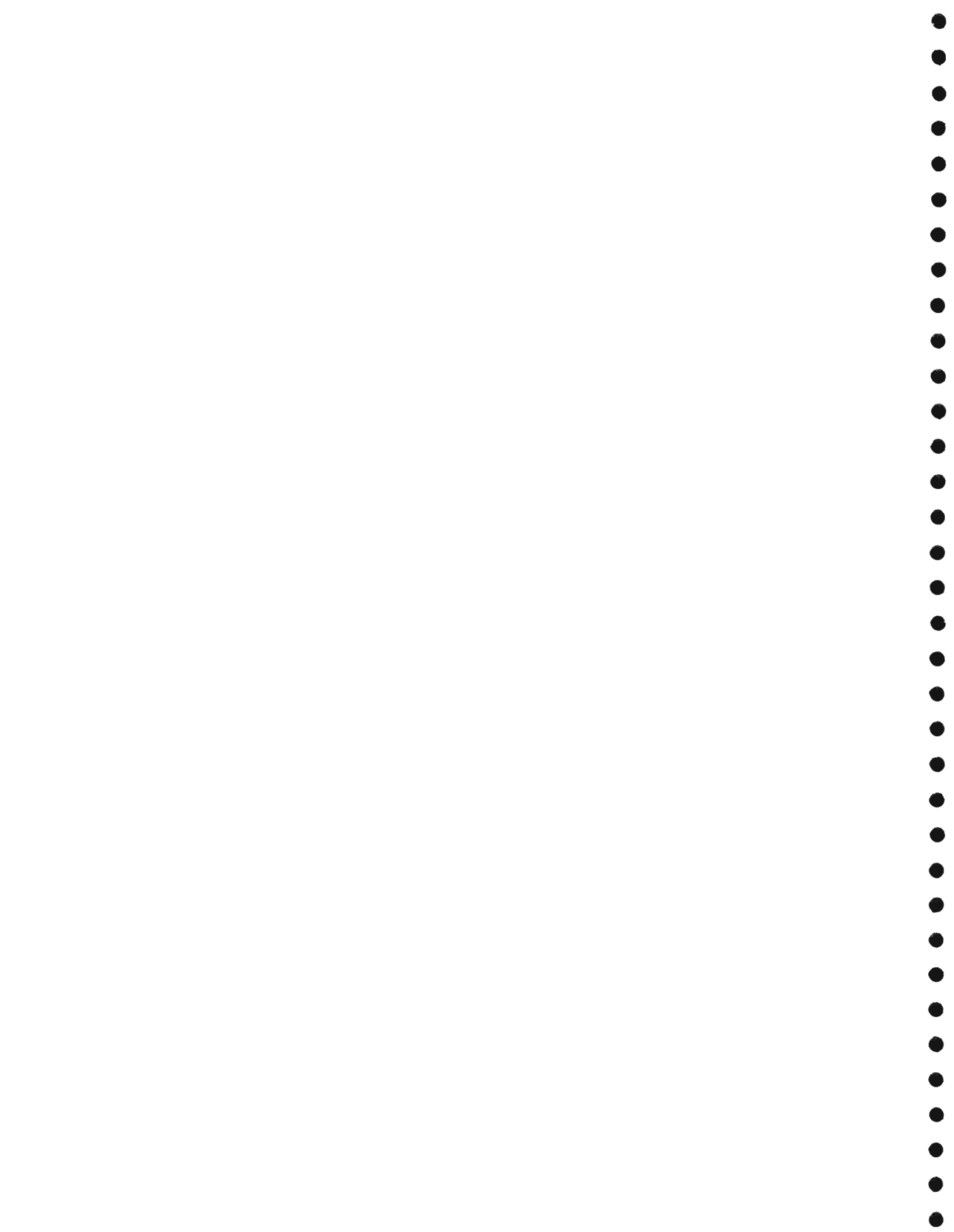
11. Enter the command to delete the CALCUSTS list.

_____<r>

12. POINTER-FILE files must have a _____ in the file defining the item in the MD.

13. Write the ACCESS sentence which sorts the catalogued list pointers in the POINTER-FILE and shows the time and date that they were saved.

> _____<r>



FORMATTING WITH DATA DEFINITIONS

7.1

INTRODUCTION

The intention of this chapter is to introduce some basic methods for effecting the output of ACCESS by altering the contents of data definition items. The main system tool used to alter these items is Editor. (Another good reason to be familiar with the EDIT verb and all of the associated commands.)

The parameters contained within a definition item are actually instructions to ACCESS for data origin, format, calculation and correlation. These parameters are dynamically processed so that when altered, the results can be immediately tested by invoking an ACCESS sentence containing the descriptive name of the definition item.

The layout of a data definition item is as follows:

ATTRIBUTE NUMBER	DESCRIPTION
000	The item-id. The descriptive name of the definition item.
001	The type of definition. A - Attribute definition. S - Synonym definition. X - Protected. The system doesn't care if a type is an A or an S. The decision between the two is totally up to the user. A protected attribute (X) causes an ACCESS request to abort if this data definition is used in a request.
002	The attribute position of the data.
003	The optional column heading. The definition item-id is used as a column header by default if 003 is null.
004	Associated attributes. This is used if attributes contain multivalued lists which are correlated. That is, they are always reported together. Don't worry about this right now. The results are too esoteric for this discussion.
005	Not used.
006	Not used.
007	A valid data conversion or mask. Performs functions at print time, the second phase of ACCESS.
008	A valid "formula" used to perform arithmetic, logical and string manipulations of one or more attributes. Performs functions at selection time, the first phase of ACCESS.
009	Justification of data in the output column. L - Left. Data wraps if wider than the field length. R - Right (used for numeric attributes). T - Text. Data wraps at a blank. U - Unconditional left. Data does not wrap.
010	The width of the output column.

Each of the attributes in a definition item is described by another definition item residing in the master dictionary (MD) of the current account. These are the descriptive names used by the LISTDICT process discussed briefly in Chapter 2. The following is a list of the MD definition items and the attribute positions that they address.

DEFINITION NAME	ATTRIBUTE POSITION
D/CODE	001
A/AMC	002
S/NAME	003
S/AMC	004
V/CONV	007
V/CORR	008
V/TYP	009
V/MAX	010

These descriptive words can be used to generate a custom report of dictionary data definitions. Try this:

```
>SORT DICT CUSTOMERS BY A/AMC BY D/CODE...
...WITH D/CODE # "D]" BREAK-ON A/AMC V/TYP V/MAX<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
DICT CUSTOMERS	filename
BY A/AMC BY D/CODE	sequence clause
WITH D/CODE # "D]"	selection clause
BREAK-ON A/AMC V/TYP V/MAX	output-list

```
>SORT DICT CUSTOMERS BY A/AMC BY D/CODE...
...WITH D/CODE # "D]" BREAK-ON A/AMC V/TYP V/MAX<r>
```

PAGE 1

10:51:02 13 DEC 1988

CUSTOMERS. A/AMC TP MAX

```
0          0 R 10
CUSTNUM    0 R 10
```

```
1          1 L 20
COMPANY    1 L 20
```

```
2          2 L 20
CONTACT    2 L 20
```

```
3          3 L 20
ADDRESS    3 L 20
```

```

4          4 L 15
CITY      4 L 15

***

5          5 L 2
STATE    5 L 2

***

6          6 L 10
ZIP      6 L 10

***

7          7 R 12
PHONE    7 L 14

***

BALANCE   8 R 12

***

DATE.DUE  9 R 12

***

***

18 ITEMS LISTED.

```

7.2 COLUMN HEADERS

As has been demonstrated, column headers default to the item-id of the definition item when attribute 003 is empty. In the section of Chapter 4 which covers implicit report formatting, the numeric data definitions are all given alpha column headers to override the numeric item-ids.

Special column headers can be added to any data definition item. As an exercise, modify the column header for the COMPANY data definition. Instead of simply having the word COMPANY as a column header, replace attribute 003 of the definition item with the string "Company Name" mixing upper and lowercase.

Enter the following:

```

>ED DICT CUSTOMERS COMPANY<r>
TOP
.G3<r>
003
.R<r>
003 Company Name
.FI
'COMPANY' FILED.
>

```

EDIT the COMPANY
data definition item.

Goto line 003.

Line 003 is empty.

Invoke the replace
command.

File and exit.

Enter this text.

Back at TCL.

Now test this change with the following sentence:

```
>LIST CUSTOMERS COMPANY<r>
```

The output is as follows:

```
>LIST CUSTOMERS COMPANY<r>

PAGE 1                      12:46:53  13 DEC 1988

CUSTOMERS. Company Name.....

106      PHINK, PHINK AND PHI
          NK
107      MONA LISA LTD
100      SHIPS R US
101      ACME CARROT
102      THE GUN STORE
103      AEROBIC DEN
104      SPEAK EASY
105      DIABOLICAL SOFTWARE

8 ITEMS LISTED.
```

Notice the change in the column header for company name.

Column header text can also be stacked. That is, there can be multiple lines of column header text. This is accomplished by using value marks (ASCII Char 253, generated by a <ctl><shift><]>) to delimit the end of each line.

Modify the COMPANY column heading to be two lines, "Company" on the first, and "Name" on the second, as follows:

```
>ED DICT CUSTOMERS COMPANY<r>
TOP
.G3<r>
003 Company Name
.R/y /y<ctl><shift><]>/<r>
003 Company]Name
.FI
'COMPANY' FILED
>
```

EDIT the definition item.

Goto line 003.

Displayed by EDIT.

Use REPLACE to insert the value mark.

File and exit.

Back at TCL.

Now, enter the same ACCESS sentence and notice the result.

```
>LIST CUSTOMERS COMPANY<r>
```

```
PAGE 1
```

```
12:53:58 13 DEC 1988
```

```
CUSTOMERS. Company.....  
Name
```

```
106      PHINK, PHINK AND PHI  
          NK  
107      MONA LISA LTD  
100      SHIPS R US  
101      ACME CARROT  
102      THE GUN STORE  
103      AEROBIC DEN  
104      SPEAK EASY  
105      DIABOLICAL SOFTWARE
```

```
8 ITEMS LISTED.
```

Finally, column headers can be completely removed by placing a back-slash (/) in attribute 003 of the definition item. This time, however, create a synonym definition to use so that the original remains unchanged. To start off, COPY the item COMPANY to a new item-id, CO as follows:

```
>COPY DICT CUSTOMERS COMPANY<r>
```

```
TO:CO<r>
```

```
1 COMPANY TO CO
```

Now, modify the CO definition as follows:

```
>ED DICT CUSTOMERS CO<r>
```

```
TOP
```

```
.G1
```

```
001 A
```

```
.R/A/S<r>
```

```
001 S
```

```
.G3<r>
```

```
003 Company]Name
```

```
.R<r>
```

```
003 \<r>
```

```
.FI<r>
```

```
'CO' FILED
```

Goto attribute 001.

Replace the A with S, designating a synonym. This is not required, but helps you keep track of the synonym definitions. Remember, the system doesn't care if you use an A or an S.

Goto attribute 003.

Replace the entire attribute.

Enter a back-slash and the <return>.

File and exit.

Now enter the following and observe the results:

```
>LIST CUSTOMERS CO<r>

PAGE 1                      13:43:19  13 DEC 1988

CUSTOMERS. ....

106      PHINK, PHINK AND PHI
          NK
107      MONA LISA LTD
100      SHIPS R US
101      ACME CARROT
102      THE GUN STORE
103      AEROBIC DEN
104      SPEAK EASY
105      DIABOLICAL SOFTWARE

8 ITEMS LISTED.
```

7.3

JUSTIFICATION

Justification of the data within the column can be altered by changing parameters in the data definition item at attribute 009. Create another synonym for COMPANY called COMP.

```
>COPY DICT CUSTOMERS COMPANY<r>
TO:COMP<r>
```

1 COMPANY TO COMP

System message.

Now, you can attack the problem of customer 106, PHINK, PHINK and PHINK whose company name keeps wrapping in the column.

One way to attack this is to request an unconditional (U) left justification. This states that, no matter what the actual length, do not wrap the result. Change the justification of COMP from L to U as follows:

```
>ED DICT CUSTOMERS COMP<r>
TOP
.G1
001 A
.R/A/S<r>
001 S
.G9<r>
009 L
.R/L/U<r>
.FI<r>
'COMP' FILED.
```

Goto attribute 001,
the definition type.

Make it an S type by
replacing the A with an S.

Goto attribute 009,
the justification.

Replace L with U.

File and exit.

Now enter the following command sentence to list only customer 106:

```
>LIST CUSTOMERS '106' COMP CONTACT<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
'106'	item-list
COMP CONTACT	output-list

```
>LIST CUSTOMERS '106' COMP CONTACT<r>
```

```
PAGE 1                                13:04:57  13 DEC 1988

CUSTOMERS. Company..... CONTACT.....
           Name

106        PHINK, PHINK AND PHINPHINK, CLYDE

END OF LIST
```

Notice that a company name that's too long runs into the next column. Not very desirable.

Another way to alleviate this problem is to use the text (T) justification parameter. As an exercise, change the U on attribute 009 of COMP to a T. Now, enter the same sentence as above:

```
>LIST CUSTOMERS '106' COMP CONTACT<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
'106'	item-list
COMP CONTACT	output-list

```
>LIST CUSTOMERS '106' COMP CONTACT<r>
```

```
PAGE 1                                13:09:58  13 DEC 1988

CUSTOMERS. Company..... CONTACT.....
           Name

106        PHINK, PHINK AND      PHINK, CLYDE
           PHINK

END OF LIST
```

Notice that the T justification indicates that column wrapping should be performed at a space and not in the middle of a word. Text (T) justification is quite useful for columnar output of comments.

A final word about justification. The justification parameter also controls SORTed order if the definition name is used in a BY clause. L, T, and U cause a sort left justified and R causes a sort right justified. It should be noted that numeric fields (money, etc.) should always be right justified.

For example, if an attribute containing numbers ranging from 1 to 1,000 is sorted left justified, the output occurs in this order:

1 10 100 100 11 110 1100 etc.

If right justified, the sort is in correct numeric sequence:

1 2 3 4 5 6 7 8 9 10 11 etc.

7.4

LENGTH

The length field in attribute 010 can be modified to extend the width of the output column. Note: extending column width may make the total width of the report greater than the current width setting in the TERM statement. Always test any changes.

Fix the problem with COMPANY, once and for all. Change the column width from 20 to 25 as follows:

```
>ED DICT CUSTOMERS COMPANY<r>
TOP
.G10<r>
010 20
.R/0/5<r>
010 25
.FI
'COMPANY' FILED
><
```

Now enter the same ACCESS sentence as before:

```
>LIST CUSTOMERS '106' COMPANY CONTACT<r>

PAGE 1                                13:24:45  13 DEC 1988

CUSTOMERS. Company..... CONTACT.....
          Name

106      PHINK, PHINK AND PHINK    PHINK, CLYDE

END OF LIST
```

Notice that the company name no longer wraps.

HIDDEN COLUMNS

Columns can also be "hidden" on a report, even though the definition name is specified in the output-list of an ACCESS command sentence. This is desirable when a field is being used for a SORT and BREAK-ON with the break value being displayed at the break line via the 'V' BREAK-ON option. It is redundant to also show the columnar details for the break value.

A variation of the following command sentence is to be used:

```
>SORT CUSTOMERS BY STATE COMPANY BREAK-ON STATE "'V'" TOTAL BALANCE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
COMPANY BREAK-ON STATE...	
... "'V'" TOTAL BALANCE	output-list

The BREAK-ON option 'V' is used to display the break value at the break line. This generates the following report:

```
>SORT CUSTOMERS BY STATE COMPANY BREAK-ON STATE "'V'" TOTAL BALANCE<r>
```

PAGE 1

13:49:36 13 DEC 1988

CUSTOMERS.	Company.....	STATE	Balance Due.
	Name		

102	THE GUN STORE	CA	12.34
103	AEROBIC DEN	CA	89.12
105	DIABOLICAL SOFTWARE	CA	4.56
106	PHINK, PHINK AND PHINK	CA	126.75
107	MONA LISA LTD	CA	789.32

CA	1022.09
----	---------

100	SHIPS R US	NY	567.80
101	ACME CARROT	NY	56.75

NY	624.55
----	--------

104	SPEAK EASY	PA	23.42
-----	------------	----	-------

PA	23.42
----	-------

***	1670.06
-----	---------

8 ITEMS LISTED.

To eliminate the redundant reporting of the STATE at each detail line, the state output column must be hidden. A column is hidden by modifying two parameters in the definition item:

- The columnar header is suppressed by placing a back-slash (\) in attribute 003 of the definition item.
- Use a zero (0) value for the column width in attribute 010.

First, create a synonym for STATE called HIDE.STATE as follows:

```
>COPY DICT CUSTOMERS STATE<r>
TO:HIDE.STATE<r>

1 STATE TO HIDE.STATE ← System message.
```

Now, as an exercise, modify the HIDE.STATE definition item to look as follows:

```
ID: HIDE.STATE
001 S
002 5
003 \
004
005
006
007
008
009 L
010 0
```

The previous command sentence can be rewritten as:

```
>SORT CUSTOMERS BY STATE COMPANY BREAK-ON HIDE.STATE "'V'" TOTAL BALANCE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
COMPANY BREAK-ON HIDE.STATE...	
... "'V'" TOTAL BALANCE	output-list

Notice that the report is still being sequenced by STATE, but the output column is addressed with the new synonym, HIDE.STATE.

The report appears as follows:

>SORT CUSTOMERS BY STATE COMPANY BREAK-ON...

...HIDE.STATE "V" TOTAL BALANCE<r>

PAGE 1

13:56:18 13 DEC 1988

CUSTOMERS. Company..... Balance Due.
Name

102	THE GUN STORE	12.34
103	AEROBIC DEN	89.12
105	DIABOLICAL SOFTWARE	4.56
106	PHINK, PHINK AND PHINK	126.75
107	MONA LISA LTD	789.32

CA 1022.09

100	SHIPS R US	567.80
101	ACME CARROT	56.75

NY 624.55

104	SPEAK EASY	23.42
-----	------------	-------

PA 23.42

*** 1670.06

8 ITEMS LISTED.

EXERCISE

7

1. Fill in the attribute position next to each description of an entry in a data definition item.

_____ Justification. L, R, T or U.

_____ A conversion or correlative performed in the third phase of ACCESS, print time.

_____ The attribute position (amc) of the data.

_____ The maximum width of the output column.

_____ The definition type code. A, S or X.

_____ A conversion or correlative performed during the second phase of ACCESS, selection time.
2. Write an ACCESS statement to list all synonym (S) type data definitions found in the dictionary of CUSTOMERS.

> _____ <r>
3. What keystrokes are used to form a multiple line column header?

4. Match the following:

_____ L a. Indicates a null column header.
_____ A b. Unconditional left. The data in the column is not wrapped.
_____ S c. Right justified output.
_____ T d. An attribute definition type.
_____ \ e. A synonym definition type.
_____ R f. Protected definition.
_____ U g. Left justified. The data is wrapped in the output column.
_____ X h. Text justification. The data is wrapped on a space.

5. Fill in the blanks for the following data definition item describing a left justified comment field held in attribute 15 of the data item.

ID: COMMENTS

001 A

002 _____

003

004

005

006

007

008

009 _____

010 10

6. The output of the COMMENTS field looks like this:

COMMENTS..

THIS IS TH

E ONLY COM

MENT FIELD

IN THE IT

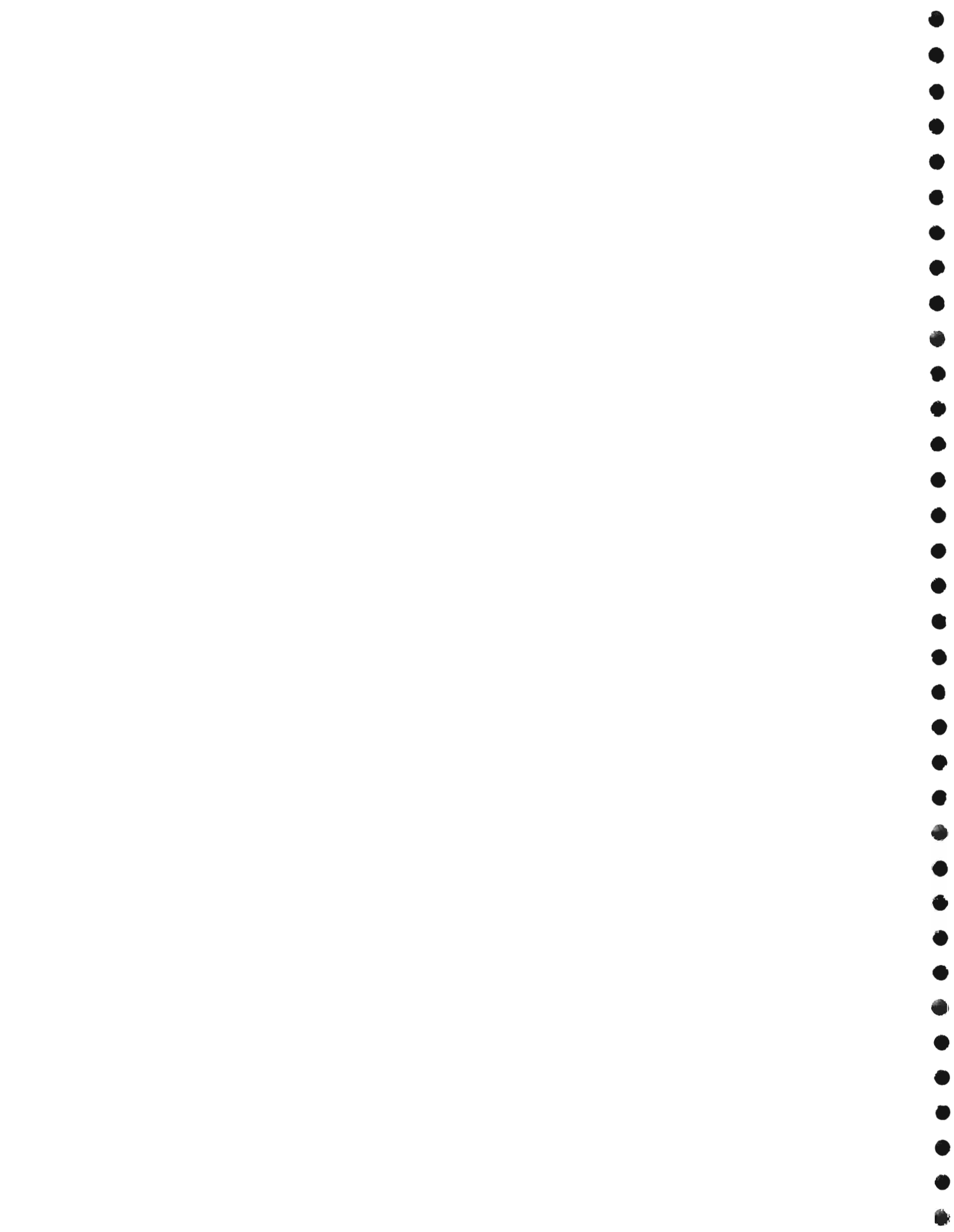
EM.

Name two changes that can be made to the COMMENTS definition item to make the output more readable.

7. Columns can be hidden on output by doing two things to the data definition item.

Place a _____ in attribute _____.

Place a _____ in attribute _____.



8

CONVERSIONS AND CORRELATIVES

8.1

OVERVIEW

Here is where things can get complicated. Don't fear. This chapter covers only the most commonly used conversions and correlatives, and how they relate directly to the sample CUSTOMERS file. Not every possible combination is covered. Nor is every nuance of "programming" in ACCESS. Such an endeavor would require an entire workbook in itself. (This is on the horizon.) However, it is important to get some basic concepts laid out.

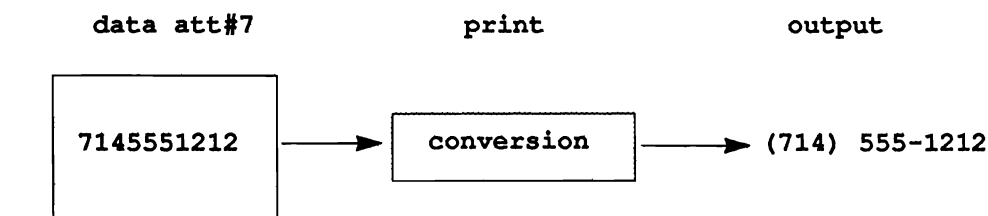
Conversions and correlatives are the parameters which reside in either attributes 007 or 008 of the data definition item.

Conversions are used to alter the format of the data stored in an attribute so that it is more easily interpreted on a report. Data as it exists in an item is said to be in internal format. That is, the data is in its most unencumbered and compact form. Conversions allow the data to be presented in external or output format. This is a form more easily understood by human beings.

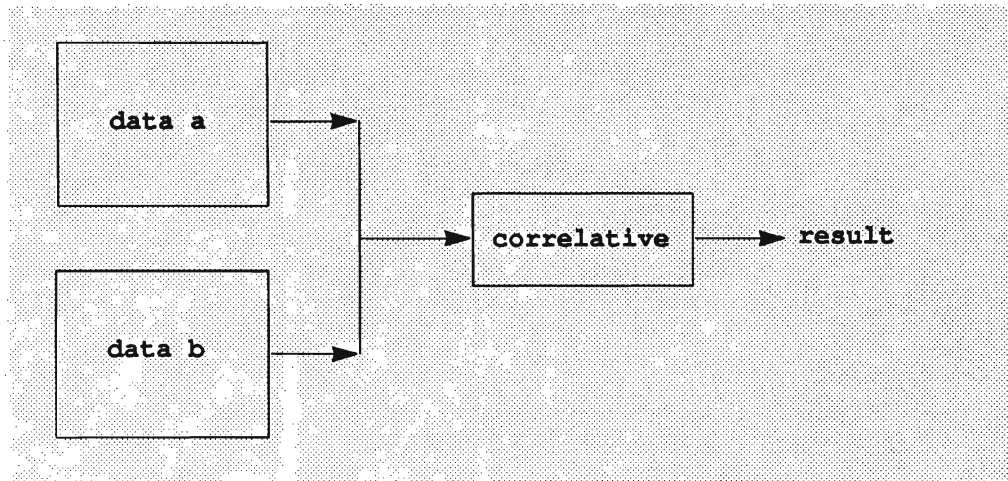
For example, the phone number attribute in the CUSTOMERS file is a 10 digit phone number that contains no logical breaks between the area code, the prefix and the extension. To make this data more readable on a report, it requires a conversion mask so that these elements of the data are easily recognized.

The phone number 7145551212 should be displayed as (714) 555-1212.

The conversion mask (the actual mask parameter is discussed later in this section) reformats the data without changing the stored data.



Correlatives are "formulas" which perform arithmetic, logical or string building functions on the data. Whereas conversions simply reformat a single data element for readability, correlatives extract and combine single or multiple data elements to produce a "virtual" data field. The word "virtual" is used to describe data which is not actually stored in an item, but is "created" via the dynamic process indicated in the correlative expression.



The data correlated can be retrieved from any attribute in the same item or in another item, or the data can be system provided. For example, the number of days that payment is overdue can be calculated by subtracting the date due from today's date. Today's date is provided by the system. (Hopefully, the correct date has been set.)

today's date - date due

(Again, the actual formula is shown later.) This performs an arithmetic calculation based on two separate elements of data to produce a new "virtual" data element which automatically changes as time passes.

When using a selection clause with a test value based on the results of a correlative, it is important to note that the placement of the correlative in attribute 007 versus 008 can affect the outcome of the report. As the data is retrieved from an item and is tested, the process dictated by the correlative in attribute 008 is performed. However, correlatives placed in attribute 007 are not performed until print time, when the selection decisions have already been made.

```
>LIST CUSTOMERS WITH OVERDUE >= "30"
```

A calculation placed in attribute 008 of the overdue definition would cause the command sentence above to perform correctly. The same calculation in attribute 007 would cause the command sentence to fail.

This is a concern for correlatives only. Simple conversions can reside in attribute 007 with no problems.

Well, the only way to really start understanding how to use conversions and correlatives is by example. So, without further adieu:

8.2

DATE AND TIME CONVERSIONS

8.2.1

The D Conversion

Dates are stored in an internal format, foreign to any traditional method of date storage. Instead of the usual month, day and year, an internally formatted date is stored as a count of days beginning at an artificial day zero.

In the PICK system, day zero is December 31, 1967. Day 1 is January 1, 1968, and day -1 is December 30, 1967. Days prior to day zero are negative and days following day zero are positive.

Why such a strange standard? I'll answer a question with a question. Have you ever tried to sort a date field stored in a "traditional" format? (BY YEAR BY MONTH BY DAY.) Worse yet, have you ever tried to figure a net 30 payment within a leap year? Well, the date routines in the PICK system make these tasks elementary.

The date conversion (D) converts an internal date to a variety of external formats.

The simple form is:

D{y}{c}

The "y" and "c" parameters are optional and have the following meaning:

PARAMETER	MEANING
y	The number of digits in the output year (0, 2 or 4). The absence of this parameter indicates a 4 digit year.
c	A single alpha character to use as a delimiter between the month, day and year. It can also be: D output the day of the month only J julian date M output the month only Q output the quarter W output the day of the week Y output the year only

The stored data values for DATE.DUE are numbers like:

8888 This is over 24 years after December 31, 1967.
-500 This is over a year before December 31, 1967.

Remember the DATE.DUE data definition?

```
ID: DATE.DUE
001 A
002 9
003
004
005
006
007 D
008
009 R
010 12
```

Notice that the date conversion "D" is placed on attribute 007 and, therefore, it is performed at print time. Attribute 007 is the usual home for data conversions.

Now, enter the following sentence and compare the results to the data actually stored in attribute 009 (using *A9) of each data item.

```
>SORT CUSTOMERS BY DATE.DUE COMPANY DATE.DUE *A9<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY DATE.DUE	sequence clause
COMPANY DATE.DUE *A9	output-list

```
>SORT CUSTOMERS BY DATE.DUE COMPANY DATE.DUE *A9<r>
```

```
PAGE 1
```

```
10:47:58 14 DEC 1988
```

```
CUSTOMERS. Company..... DATE.DUE.... *A9.....
           Name
```

105	DIABOLICAL SOFTWARE	05 APR 1965	-1000
102	THE GUN STORE	18 AUG 1966	-500
101	ACME CARROT	20 DEC 1988	7660
100	SHIPS R US	05 AUG 1989	7888
103	AEROBIC DEN	24 NOV 1989	7999
107	MONA LISA LTD	22 MAY 1991	8543
104	SPEAK EASY	03 JUN 1991	8555
106	PHINK, PHINK AND PHINK	01 MAY 1992	8888

```
8 ITEMS LISTED.
```

The R in attribute 009 of the data definition insures that the numerics are kept in proper sequence. Notice the format of the simple D conversion. Even though the date is stored in internal format, and the conversion is specified at print time, the external form can still be used in a selection clause as follows:

```
>SORT CUSTOMERS BY DATE.DUE IF DATE.DUE AFTER "01/01/80" COMPANY DATE.DUE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY DATE.DUE	sequence clause
IF DATE.DUE AFTER "01/01/80"	selection clause
COMPANY DATE.DUE	output-list

```
>SORT CUSTOMERS BY DATE.DUE IF DATE.DUE AFTER "01/01/80" COMPANY DATE.DUE<r>
```

```
PAGE 1
```

```
10:33:53 14 DEC 1988
```

```
CUSTOMERS. Company..... DATE.DUE....
             Name
```

```
105          DIABOLICAL SOFTWARE      05 APR 1965
102          THE GUN STORE             18 AUG 1966
```

```
2 ITEMS LISTED.
```

Simple conversions in attribute 007 are a two-way street. When the ACCESS sentence is interpreted, the conversion is used to convert the selection value ("01/01/80") to the equivalent internal format for comparison to the stored data. On output, the stored data is converted to the equivalent output format.

Notice that the selection clause is specifying the date in a D2/ format, even though the conversion in DATE.DUE is simply a D. The date conversion is intelligent enough to handle this.

Here are the results of different combinations of D conversion acting on the internal date -1000.

Internal Date: -1000

CONVERSION	RESULT
D	05 APR 1965
D0	05 APR
D/	04/05/1965
D2	05 APR 65
D2-	04-05-65
DO/	04/05
DD	5
DM	4 (numeric month)
DMA	APRIL (alphabetic month)
DW	1 (numeric day of the week)
DWA	MONDAY (alphabetic day of the week)
DY	1965
D2Y	65
DJ	95
DQ	2

As an exercise, replace attribute 007 of DATE.DUE with each of these and test the result.

8.2.2

The MT Conversion

Time is also stored as a sequential number instead of the cumbersome base 60 system of hours, minutes and seconds. In fact, time is stored as the number of seconds since midnight.

The MT conversion handles time and follows the format:

MT{H}{S}

Normally, the MT conversion displays a 24 hour clock. The H and S parameters are optional and specify the following:

H 12-hour clock, AM and PM.
S Output seconds.

Here are some variations of the mask time (MT) conversion for the internal time of 67000 seconds.

Internal time: 67000

CONVERSION	RESULT
MT	18:36
MTS	18:36:40
MTH	06:36PM
MTHS	06:36:40PM

CHARACTER MASKING

A string is any series of data characters. All data stored in PICK is a string. String masking allows any string to be reformatted for output.

8.3.1

The MC Conversion

The first string mask conversion is the (MC) mask character which follows the general form:

MC{/c

The slash is optional and means "not." Using the variations on the "c" code, the mask character conversions can be any of the following:

To mask out (remove) only certain character types:

- MCA Output only the alphabetic characters.
- MC/A Output all but the alphabetic characters.
- MCN Output only the numerics, (0-9).
- MC/N Output all but the numerics.

To change the case of the alphabetic characters:

- MCL Change all alphabetics to lowercase.
- MCU Change all alphabetics to uppercase.
- MCT Also known as "mask character text." Capitalize the first character of every word and convert the remaining characters to lowercase.

Other uses: (these are FYI only)

- MCP Convert all control <ctl> non-display characters to periods. This is quite helpful in weeding out garbage in attributes.
- MCDX Convert decimal values to the equivalent hexadecimal.
- MCXD Convert hexadecimal values to the equivalent decimal value.

Here is what each of these conversions produces when acting upon the following address: 123 MAIN street Apt. 2-a

CONVERSION	RESULT
MCA	MAINstreetApta
MC/A	123 . 1-
MCN	1231
MC/N	MAIN street Apt. -a
MCL	123 main street apt. 1-a
MCU	123 MAIN STREET APT. 1-A
MCT	123 Main Street Apt. 1-A

Alter the ADDRESS definition to display the address in a text format mixing upper and lower-case as follows:

```
>ED DICT CUSTOMERS ADDRESS<r>
TOP
.G7<r>
007
.R//MCT<r>
007 MCT
.FI
'ADDRESS' FILED
>
```

Replace the null attribute with the MCT conversion.

File and exit.

Back at TCL

Now check the results by entering:

```
>LIST CUSTOMERS CONTACT ADDRESS<r>

PAGE 1                                11:15:02  14 DEC 1988

CUSTOMERS. CONTACT..... ADDRESS.....

106      PHINK, CLYDE      1600 Pennsylvania Ave
107      DAVINCI, LEONARDO 15 Century Dr
100      HORNBLOWER, HORATIO 1110-1 W 57Th St
101      BUNNY, BUGS      10 Hole-In-Ground
102      FUDD, ELMER      30 Wabbit Way
103      PORCELANA, PORCHE 139 Hollywood Blvd
104      PARTICIPLE, DANGLY 3 Dat Dare St
105      GROG, BILL SPIDER 01100 Binary Way

8 ITEMS LISTED.
```

8.4 STRING MASKING

String masking allows any string to be reformatted to match a specified pattern. This is accomplished with the mask (M) conversion following the general format:

Mjust fill length

COMMAND	DESCRIPTION
---------	-------------

just	The justification of the output string, L for left or R for right.
fill	The single character parameter used to specify the character to use to fill the full length of the output string. The fill parameter can be: # fill spaces % fill zeros * fill asterisks
length	The maximum length of the output string. Unlike the column length specification in 010, the ML and MR masks truncate the string at the "length."

Passing the data string "TEST" through this conversion results in the following:

CONVERSION	RESULT
ML#10	TEST
MR#10	TEST
ML%10	TEST000000
MR%10	000000TEST
ML*10	TEST*****
MR*10	*****TEST

The fill asterisk and fill zero are probably more effective when outputting numerics. The number 55 is used in the following example:

ML#10	55
MR#10	55
ML%10	5500000000
MR%10	0000000055
ML*10	55*****
MR*10	*****55

The fill parameter followed by the length is a "shorthand" method for indicating an output format mask. For example, ML#10 is equivalent to ML##### or ML(#####).

Any output format mask can be enclosed in parentheses as follows:

ML(explicit format)

An explicit format can be demonstrated by altering the PHONE definition item in the DICT of CUSTOMERS so that it looks like this:

```
ID: PHONE
001 A
002 7
003
004
005
006
007 ML( (###) ###-####)
008
009 L
010 12
```

The output mask, ML((###) ###-####) contains literal characters, (,), <space> and -. The external parenthesis are strictly to indicate inclusion and are not part of the output string. Every character other than a fill parameter within the enclosure, including other parentheses, is treated as a literal.

>LIST CUSTOMERS CONTACT PHONE<r>

PAGE 1

11:40:02 14 DEC 1988

CUSTOMERS. CONTACT..... PHONE.....

106	PHINK, CLYDE	(213) 555-1112
107	DAVINCI, LEONARDO	(213) 555-4444
		(213) 555-4321
100	HORNBLOWER, HORATIO	(212) 555-7676
		(212) 555-6666
101	BUNNY, BUGS	(212) 555-6666
		(212) 555-6661
102	FUDD, ELMER	(213) 555-9999
103	PORCELANA, PORCHE	(213) 555-7777
104	PARTICIPLE, DANGLY	(215) 555-4433
		(215) 555-6321
		(215) 555-4444
105	GROG, BILL SPIDER	(408) 555-4444

8 ITEMS LISTED.

Notice that the conversion is automatically repeated for each multivalued line.

The above format mask can also be written in shorthand as follows:

ML((#3) #3-#4)

8.5 MONEY (DECIMAL) MASKING

Monetary data is usually stored without decimal points (periods) as part of the data field. This data is said to contain implied decimals. The central reason for storing money fields as implied decimals is that ACCESS only performs integer arithmetic. Decimal values are truncated. A monetary value that is stored in the customer BALANCE field has 2 implied decimal places. For example:

STORED	IMPLIED
12675	126.75
78932	789.32
10000	100.00
45	0.45

The MR or MD Conversions

Decimal masking is performed by using either the MD or MR conversions. These are interchangeable. The MR conversion has the same basic format as the previously discussed string masking conversion (Mjust) with specific extensions for formatting decimal values. The general format is:

MR{precision}{scaling}{Z}{,}{sign}{\$}fill length

COMMAND	DESCRIPTION
precision	The number of decimal places in the resulting number.
scaling	The number of places to the left to move the decimal point before rounding to the precision.
Z	Suppresses leading zeros.
	Places a comma (,) at the thousands, millions, etc., positions.
sign	A sign code which designates how to handle negative numbers. The values can be:
	C Negative numbers are followed by the credit indicator, "CR."
	D Positive numbers are followed by the debit indicator, "DB."
	E Negative numbers are enclosed in angular brackets, <number>.
	M Negative numbers are followed by the minus sign, number-.
	N The minus sign is suppressed.
\$	Prefix the answer with a dollar sign.
fill and length	The same as in the string ML and MR conversions.

If the scaling and precision match, then only one needs to be indicated. For example:

INTERNAL VALUE	CONVERSION	OUTPUT
67897	MR2	678.97

The conversion MR2 is identical to MR22. Redundant precision and scaling parameters need not be specified. This states that the decimal point is to be moved 2 places to the left and the number is to be reported with a 2 decimal precision.

When the precision and scaling are mismatched, the result is rounded. For example:

INTERNAL VALUE	CONVERSION	SCALED	ROUNDED OUTPUT
67897	MR24	6.7897	6.79
67897	MR02	678.97	679

The first example moves the decimal 4 places to the left and then rounds to a 2 place precision. The second moves the decimal 2 places to the left and reports with no decimals.

Here are a series of examples using the MR conversion.

INTERNAL VALUE	CONVERSION	OUTPUT
67897	MR2\$	\$678.97
67897	MR2D\$	\$678.97DB
-67897	MR0,\$	-\$67,897
-67897	MR2E	<678.97>
67897	MR2,\$*12	\$*****678.97
67897	MR04\$	\$7

Here is the data definition for BALANCE in the CUSTOMERS file.

```

ID: BALANCE
001 A
002 8
003 Balance Due
004
005
006
007 MR2
008
009 R
010 12

```

Create the following synonyms in the DICT of CUSTOMERS.

<pre> ID: BAL1 001 S 002 8 003 004 005 006 007 MR2, \$ 008 009 R 010 12 </pre>	<pre> ID: BAL2 001 S 002 8 003 004 005 006 007 MR02, \$ 008 009 R 010 12 </pre>	<pre> ID: BAL3 001 S 002 8 003 004 005 006 007 MR24*12 008 009 R 010 12 </pre>	<pre> ID: BAL4 001 S 002 8 003 004 005 006 007 MR2*12 008 009 R 010 12 </pre>
--	---	--	---

Now, enter the following command sentence:

```
>LIST CUSTOMERS BALANCE BAL1 BAL2 BAL3 BAL4<r>
```

PAGE 1

13:33:48 14 DEC 1988

CUSTOMERS.	Balance Due.	BAL1.....	BAL2.....	BAL3.....	BAL4.....
106	126.75	\$126.75	\$127	*****1.27	000000126.75
107	789.32	\$789.32	\$789	*****7.89	000000789.32
100	567.80	\$567.80	\$568	*****5.68	000000567.80
101	56.75	\$56.75	\$57	*****0.57	000000056.75
102	12.34	\$12.34	\$12	*****0.12	000000012.34
103	89.12	\$89.12	\$89	*****0.89	000000089.12
104	23.42	\$23.42	\$23	*****0.23	000000023.42
105	4.56	\$4.56	\$5	*****0.05	000000004.56

8 ITEMS LISTED.

As in the date and time conversions, the external format of a monetary value can be specified within an ACCESS selection clause.

```
>SORT CUSTOMERS BY BALANCE WITH BALANCE >= "100.00" CONTACT BALANCE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY BALANCE	sequence clause
WITH BALANCE >= "100.00"	selection clause
CONTACT BALANCE	output-list

```
>SORT CUSTOMERS BY BALANCE WITH BALANCE >= "100.00" CONTACT BALANCE<r>
```

PAGE 1

13:39:03 14 DEC 1988

CUSTOMERS.	CONTACT.....	Balance Due.
106	PHINK, CLYDE	126.75
100	HORNBLOWER, HORATIO	567.80
107	DAVINCI, LEONARDO	789.32

3 ITEMS LISTED.

8.6

STRING EXTRACT CORRELATIVES

String extracts are used to remove portions of a data string. As with all correlatives, placement in attribute 008 versus attribute 007 makes a difference when testing for the result in a selection clause.

8.6.1

The Text Extract (T)

The text extract allows fixed length strings to be removed based on the character position. The general form is:

Tbegpos,length

begpos The beginning character position to start the extract.

length The number of characters to extract.

For example, using a stored string of "123456789" the text extract produces the following results:

CORRELATIVE	RESULTS
T1,1	1
T1,5	12345
T3,2	34
T3,5	34567
T9,1	9

Use the text extract to remove the area code from the phone number in attribute 7 of the data item by creating the following data definition, AREA.CODE.

```
ID: AREA.CODE
001 S
002 7
003
004
005
006
007 T1,3
008
009 L
010 3
```

The extract begins at position 1 and removes 3 characters.

Now, enter the following command sentence:

```
>LIST CUSTOMERS PHONE AREA.CODE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
PHONE AREA.CODE	output-list

```
>LIST CUSTOMERS PHONE AREA.CODE<r>
```

PAGE 1

13:56:42 14 DEC 1988

CUSTOMERS. PHONE..... AREA.CODE

106	(213) 555-1112 213
107	(213) 555-4444 213
	(213) 555-4321 213
100	(212) 555-7676 212
	(212) 555-6666 212
101	(212) 555-6666 212
	(212) 555-6661 212
102	(213) 555-9999 213
103	(213) 555-7777 213
104	(215) 555-4433 215
	(215) 555-6321 215
	(215) 555-4444 215
105	(408) 555-4444 408

8 ITEMS LISTED.

The text extract is automatically performed on the multivalues in the PHONE attribute.

Now, here is where the placement of the correlative makes a difference in the results of ACCESS.
Enter the following sentence:

```
>SORT CUSTOMERS BY CONTACT WITH AREA.CODE = "408" CONTACT PHONE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY CONTACT	sequence clause
WITH AREA.CODE = "408"	selection clause
CONTACT PHONE	output-list

The result is the system error message:

```
[401] NO ITEMS PRESENT
```

How can this be? There is at least one customer in 408. The problem is that the text extract in AREA.CODE is in attribute 007. Attribute 007 is processed at print time, AFTER the data has been selected. If the result of the extract is to be tested at selection time, it MUST reside in attribute 008 of the definition item.

Change the AREA.CODE data definition as follows:

```
>ED DICT CUSTOMERS AREA.CODE<r>
TOP
.G7<r>
007 T1,3
.R/T1,3//
007
.<r>
008
.R//T1,3<r>
008 T1,3
.FI
'AREA.CODE' FILED
>
```

Now enter the same ACCESS command sentence:

```
>SORT CUSTOMERS BY CONTACT WITH AREA.CODE = "408" CONTACT PHONE<r>

PAGE 1                                14:09:26  14 DEC 1988

CUSTOMERS. CONTACT..... PHONE.....

105          GROG, BILL SPIDER    (408) 555-4444

END OF LIST
```

8.6.2 *The Group Extract (G)*

The group extract allows data strings to be removed based on a recognizable data pattern. For example, a data string may look like this:

123*ABC*1S6

This data string has three "groups," each delimited by an asterisk (*).

The group extract allows any single or series of groups to be removed from the data string. The general format is as follows:

G{skip}delimiter segments

COMMAND	DESCRIPTION
skip	An optional parameter which indicates how many delimiters to skip before beginning the extract. Starting an extract at the beginning of the string is indicated by a 0 skip value. If the skip value is omitted, the skip defaults to 0.
delimiter	A single character which is being used to separate the groups.
segments	The number of sequential groups to extract.

Using the previous string, "123*ABC*1S6", possible group extracts are as follows:

CORRELATIVE	RESULTS
G*1	123
G0*1	123
G*2	123*ABC
G1*1	ABC
G*3	123*ABC*1S6
G2*1	1S6

The CONTACT attribute in CUSTOMERS contains a person's name in the following format:

Last,<space>First<space>Middle

The group extract can be used to address each element in the CONTACT field to separate the last and first names. The last name is set apart from the rest of the attribute by a comma (.). Therefore, a group extract which addresses a single group from the beginning of the string, using a comma as a delimiter, is expressed like this:

G0,1 or G,1

Enter the following definition item into the DICT of CUSTOMERS.

```

ID: LAST.NAME
001 S
002 2
003
004
005
006
007
008 GO,1
009 L
010 15

```

Notice that the group extract is placed in attribute 008 so that the last name can be used in a selection clause.

Extracting the first name is a little trickier than grabbing the second group delimited by a comma (.). In fact, the following group extract reports both the first name and middle initial.

G1,1

However, the comma doesn't need to be used. There is another delimiter used in this data string, a space. The first name is always separated from the comma ending the last name and the middle name by spaces. With this in mind, enter the following data definition:

```
ID: FIRST.NAME
001 S
002 2
003
004
005
006
007
008 G1 1
009 L
010 15
```

When entry of these definitions is complete, enter the following command sentence:

```
>SORT CUSTOMERS BY LAST.NAME FIRST.NAME LAST.NAME CONTACT<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY LAST.NAME	sequence clause
FIRST.NAME LAST.NAME CONTACT	output-list

```
>SORT CUSTOMERS BY LAST.NAME FIRST.NAME LAST.NAME CONTACT<r>
```

```
PAGE 1 14:34:09 14 DEC 1988
```

```
CUSTOMERS. FIRST.NAME..... LAST.NAME..... CONTACT.....
```

101	BUGS	BUNNY	BUNNY, BUGS
107	LEONARDO	DAVINCI	DAVINCI, LEONARDO
102	ELMER	FUDD	FUDD, ELMER
105	BILL	GROG	GROG, BILL SPIDER
100	HORATIO	HORNBLOWER	HORNBLOWER, HORATIO
104	DANGLY	PARTICIPLE	PARTICIPLE, DANGLY
106	CLYDE	PHINK	PHINK, CLYDE
103	PORCHE	PORCELANA	PORCELANA, PORCHE

```
8 ITEMS LISTED.
```

FILE TRANSLATE CORRELATIVE

The file translate process uses data in the current item to read data from another item in a different file. This helps to alleviate redundancy by allowing repetitive data to be retrieved from a "code lookup" file, rather than having it stored in every item.

For example, in the CUSTOMERS file, the STATE field is kept as a two alphabetic character abbreviation. The state name is kept in a "lookup" file called STATES. (STATES should have been created in Chapter 2, "Getting Started.") The state abbreviation is the item-id for each item in the STATES file. The contents of a STATES item might include the state name, region code, tax structure, etc. However, for the purposes of this example, the STATES file uses the following layout:

```
ID:  state abbreviation
001  state name
```

Now it's time to add the state code items to the STATES file. Again, you must be familiar with the Editor to be able to enter the following items.

```
NY
001 NEW YORK

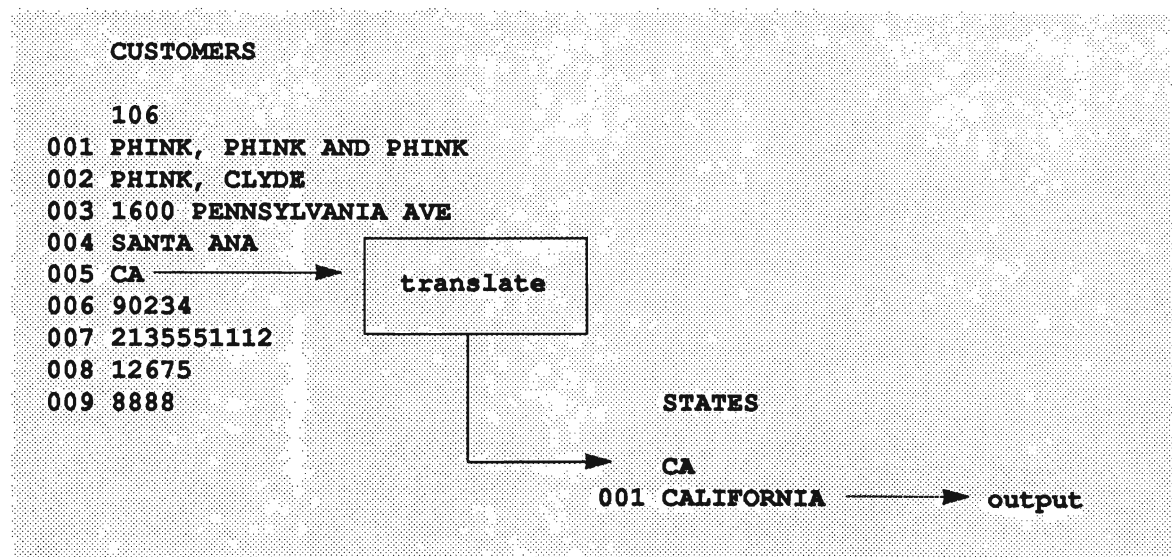
IL
001 ILLINOIS

PA
001 PENNSYLVANIA

CA
001 CALIFORNIA
```

The file translate uses the fifth attribute of a CUSTOMERS item to read the state name from the STATES file.

The STATE.NAME path:



The Translate Syntax

The general form of the file translate correlative is:

T(DICT)filename;option{n};iamc;oamc;bamc)

Here's what the parameters mean:

PARAMETER	DESCRIPTION
filename	This is the target filename. The optional DICT specification addresses the dictionary level of the target file.
option	The read option. This indicates what to do if the addresses data is there or not. The option codes can be: C If the value being read does not exist, use the original value. X If the value being read does not exist, output a null. The above options are the most frequently used. The following options are provided strictly FYI, since they are seldom used. (At least I've never seen them used. I tried to use them once, but nothing seemed to work.) V Verify existence only. The read attempt must be successful and the targeted attribute must have a value, otherwise an error message is displayed. I Input verify only. O Output verify only.
n	An optional numeric value mark count which must immediately follow the "option" code. This extracts a specific value position in the targeted attribute.
iamc	The attribute mark count to be used for the input conversion. This is seldom used, and can remain null.
oamc	The output attribute mark count. This is the attribute to be passed back through the translation.
bamc	The optional break attribute mark count. The contents of this attribute are displayed at the break line (BREAK-ON must be used in the command sentence) in place of the "oamc."

Confusing enough? When all of this is said and done, the only parameters that you have to deal with are "filename," "option" and "oamc." The rest of the parameters can remain null.

The data definition, STATE.NAME, should be added to the DICT of CUSTOMERS.

ID: STATE .NAME

001 S

002 5

003

004

005

006

007

008 TSTATES;X;;1

009 L

010 15

A synonym for
STATE.

The state abbreviation is in
attribute 5 of the data item.

The translate correlative.
filename = STATES
option = X
oamc = 1

This definition uses attribute 5 as an item-id to read the first attribute of the STATES file. If the read is not successful, the output is null.

Now enter the following ACCESS command sentence:

```
>LIST CUSTOMERS STATE STATE.NAME<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
STATE STATE.NAME	output-list

```
>LIST CUSTOMERS STATE STATE.NAME<r>
```

```
PAGE 1                                09:47:09  15 DEC 1988
```

```
CUSTOMERS. STATE STATE.NAME.....
```

106	CA	CALIFORNIA
107	CA	CALIFORNIA
100	NY	NEW YORK
101	NY	NEW YORK
102	CA	CALIFORNIA
103	CA	CALIFORNIA
104	PA	PENNSYLVANIA
105	CA	CALIFORNIA

```
8 ITEMS LISTED.
```

As in all correlatives, if the full name needs to be tested in a selection clause, the translate correlative must be placed in attribute 008. Otherwise, the translate can reside in attribute 007.

8.7.2

A Hypothetical Case

A more extensive implementation of a code file lookup involves using the zip code to imply the city and state. Therefore, the zip is the only data element required in each item of the NEW.CUSTOMERS file. All of the possible zip codes are kept as item-ids in a special ZIPS file.

Sample item layouts:

```
NEW.CUSTOMERS file
```

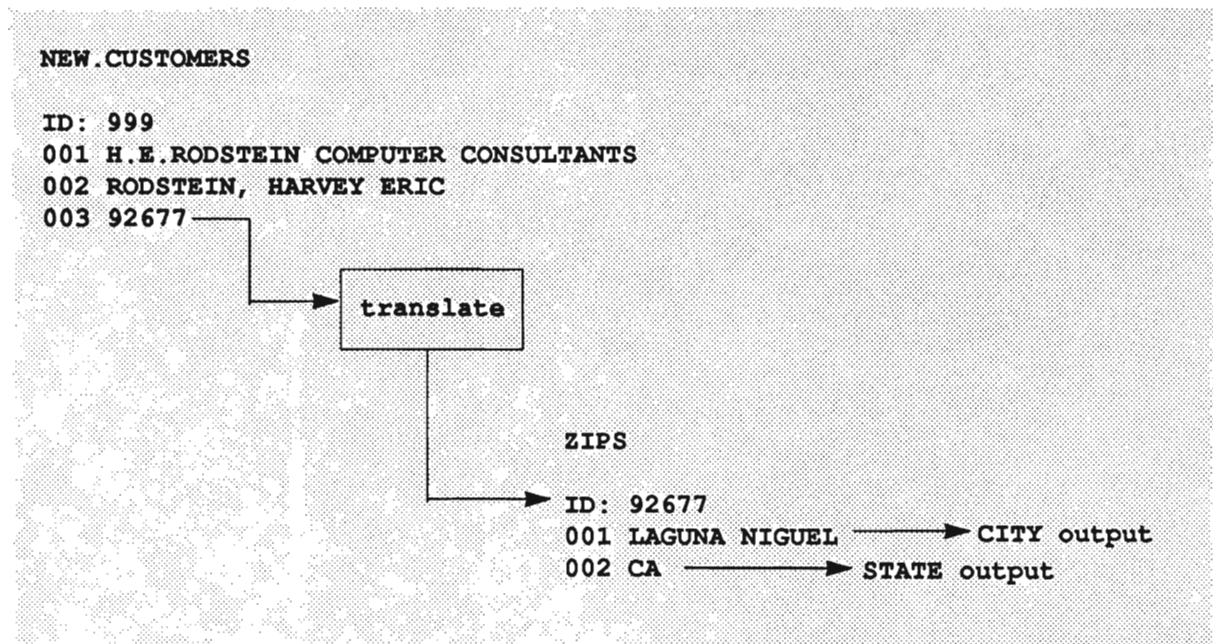
```
ID: customer number  
001 Company  
002 Contact  
003 Zip
```

```
ZIPS file
```

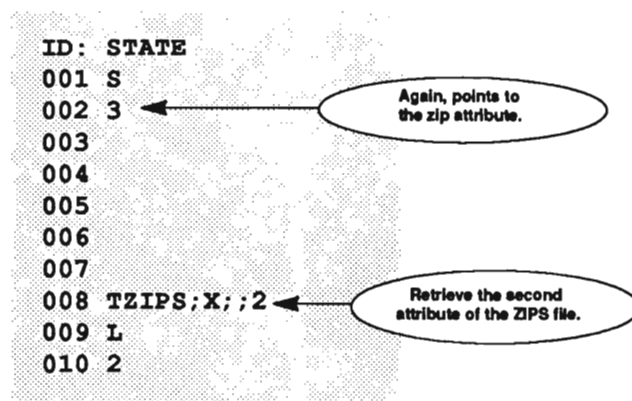
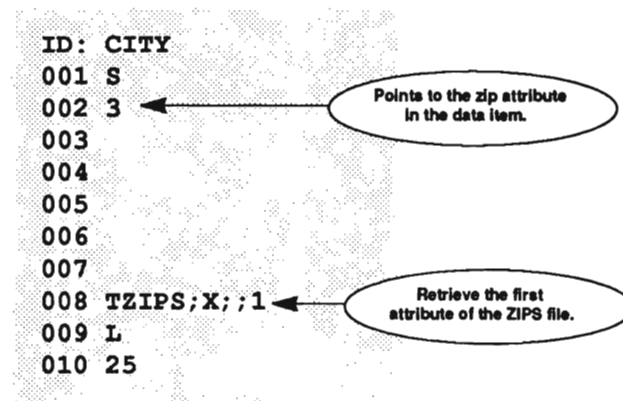
```
ID: zipcode  
001 city  
002 state
```

The two attributes contained in the ZIPS file are the city and state. The CITY and STATE data definition items in the dictionary of the NEW.CUSTOMERS file point to the zip code attribute and translate to the ZIPS file to retrieve the data.

The CITY and STATE path:



The following are the definition items which perform the described translation.



8.8

MULTIPLE CONVERSIONS AND CORRELATIVES

Multiple conversions and correlatives are used in a definition item to do multi-stage processing of a data element. Staged processing is specified by delimiting each conversion and correlative with value marks. The results of an entry in the multivalued list are passed to the subsequent entries in the order of specification.

The general form is:

Conv]Conv]Conv]...

The "Conv" represents any of the aforementioned conversions and correlatives. The right bracket (]) represents a value mark. (Keyboard generated by a <ctl><shift><]>.)

For example, the definition LAST.NAME uses a group extract to remove the last name from the CONTACT attribute. Once the last name is extracted, it can be processed by a series of correlatives or conversions.

Modify the LAST.NAME definition item so that it looks like this:

LAST . NAME

```

001 S
002 2
003
004
005
006
007 G0,1]MCT
008
009 L
010 15
  
```

First, do the group extract and then mask the characters in a text format. (The bracket is a value mark.)

Test this with the following command sentence:

```
>SORT CUSTOMERS BY CONTACT CONTACT LAST.NAME<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY CONTACT	sequence clause
CONTACT LAST.NAME	output-list

```
>SORT CUSTOMERS BY CONTACT CONTACT LAST.NAME<r>
```

```
PAGE 1
```

```
10:18:35 15 DEC 1988
```

```
CUSTOMERS. CONTACT..... LAST.NAME.....
```

101	BUNNY, BUGS	Bunny
107	DAVINCI, LEONARDO	Davinci
102	FUDD, ELMER	Fudd
105	GROG, BILL SPIDER	Grog
100	HORNBLOWER, HORATIO	Hornblower
104	PARTICIPLE, DANGLY	Participle
106	PHINK, CLYDE	Phink
103	PORCELANA, PORCHE	Porcelana

```
8 ITEMS LISTED.
```

Do the same thing for the STATE.NAME, but this time mask all of the characters to lowercase (MCL). Modify the STATE.NAME definition as follows:

```
ID: STATE.NAME
001 S
002 5
003
004
005
006
007
008 TSTATES;X;;1]MCL
009 L
010 15
```

Use this definition in the following sentence:

```
>LIST CUSTOMERS STATE STATE.NAME<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
STATE STATE.NAME	output-list

```
>LIST CUSTOMERS STATE STATE.NAME<r>
```

```
PAGE 1
```

```
09:47:09 15 DEC 1988
```

```
CUSTOMERS. STATE STATE.NAME.....
```

106	CA	california
107	CA	california
100	NY	new york
101	NY	new york
102	CA	california
103	CA	california
104	PA	pennsylvania
105	CA	california

```
8 ITEMS LISTED.
```

All of the state names are forced to lowercase.

Conversions and correlatives don't need to be staged in the same attribute. Since attribute 008 is processed before attribute 007, the STATE.NAME definition could also be written:

```
ID: STATE.NAME
001 S
002 5
003
004
005
006
007 MCL
008 TSTATES;X;;1
009 L
010 15
```

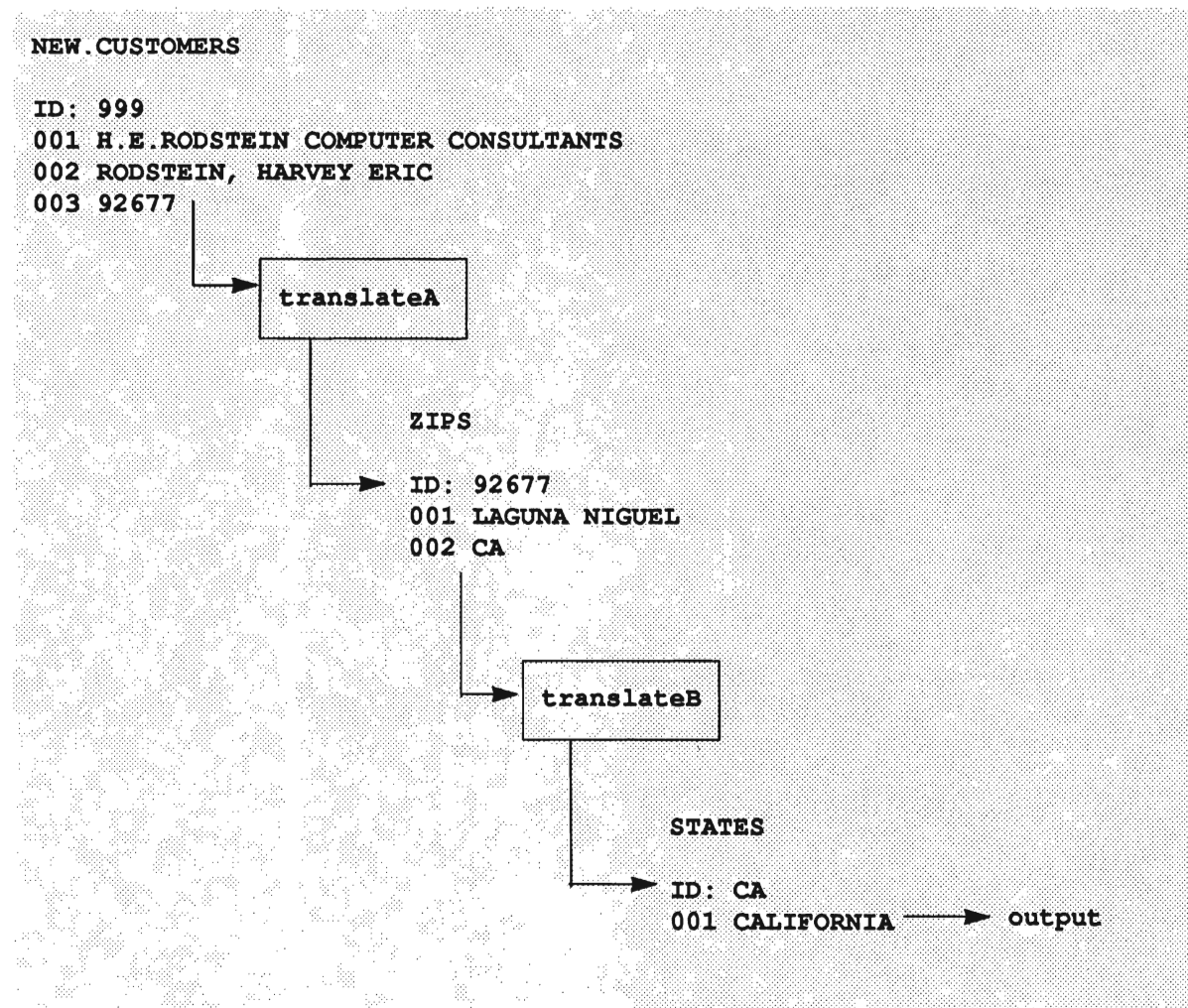
More On the Hypothetical Case

File translates can be staged to first read from one file and subsequently use that data value to read from another file. The lookup paths are virtually limitless.

```
translateA]translateB].....
```

As has been described, the zip code in NEW.CUSTOMERS is used to read the item containing the CITY and STATE from the ZIPS file. Since the STATE attribute in the ZIPS file contains a state abbreviation, a second level of translate is required to retrieve the STATE.NAME from the STATES file.

The STATE.NAME path:



The STATE.NAME definition item in DICT NEW.CUSTOMERS:

ID: STATE.NAME
001 S
002 3
003
004
005
006
007 MCT
008 TZIPS;X; ;2]TSTATES;X; ;1
009 L
010 15

The bracket is a value mark.

```
graph TD; A[STATE.NAME] --> B[001 S]; B --> C[002 3]; C --> D[003]; D --> E[004]; E --> F[005]; F --> G[006]; G --> H[007 MCT]; H --> I[008 TZIPS;X; ;2]TSTATES;X; ;1]; I --> J[009 L]; J --> K[010 15];
```

Notice that the state name is passed through a mask character text (MCT) conversion at print time.

8.9

CONCATENATION CORRELATIVE

Concatenation is the process of tagging data strings onto the end of other data strings to produce new, longer, data strings. The general form of the concatenate (C) correlative is:

C;parameter;parameter;...

The parameter values can be:

amc An attribute mark count. This retrieves the contents of the indicated attribute number.

"literal" A literal data string enclosed in quotes.

Each parameter is concatenated to the end of the previously specified parameter.

The next exercise is to build a new data definition which concatenates the CITY, STATE and ZIP attributes to produce a single string of the pattern:

city, state zip

The definition item named CITSTATZIP is added to the DICT of CUSTOMERS.

```

ID: CITSTATZIP
001 S
002 0
003
004
005
006
007 MCT
008 C;4;" , ";5;" ";6
009 T
010 45
  
```

The amc is a dummy value in this case. See below.

The concatenate correlative is the first correlative covered which combines multiple data elements to produce a compound result. In such a case, attribute 002 of the data definition becomes irrelevant for controlling the location of data origin. Therefore, the zero in attribute 002 serves as a "dummy" place holder.

The concatenation correlative appends a "," to the end of attribute 4, then appends attribute 5 to that, a space " " to that, and finally, attribute 6. Enter the following command sentence to see the results. The resulting string is then passed through a MCT to display the data in text format.

Enter the following command sentence:

```
>LIST CUSTOMERS CONTACT CITSTATZIP<r>
```

PAGE 1

12:35:08 15 DEC 1988

CUSTOMERS. CONTACT..... CITSTATZIP.....

106	PHINK, CLYDE	Santa Ana, Ca 90234
107	DAVINCI, LEONARDO	Venice Beach, Ca 97771
100	HORNBLOWER, HORATIO	New York, Ny 10019
101	BUNNY, BUGS	Brooklyn, Ny 10090
102	FUDD, ELMER	Wedondo Beach, Ca 91234
103	PORCELANA, PORCHE	Hollywood, Ca 91329
104	PARTICIPLE, DANGLY	Grammar City, Pa 12345
105	GROG, BILL SPIDER	Byteland, Ca 90001

8 ITEMS LISTED.

As an independent exercise, use CITSTATZIP in a mailing label.

8.10

FUNCTION CORRELATIVE

The function (F) correlative allows arithmetic, relational, and string operations to be performed. Since this workbook is geared to an introductory level, the arithmetic operations serve as the focal point of the discussion.

All calculations are performed in a stack. Stack processing uses a concept which can be likened to those old spring-loaded coin dispensers, where each new coin added pushes down the rest of the stack of coins. The last coin added to the stack is the first coin to be removed. As each coin is removed, the remaining coins "pop up" to fill the vacant space.

8.10.1

The F Correlative Syntax

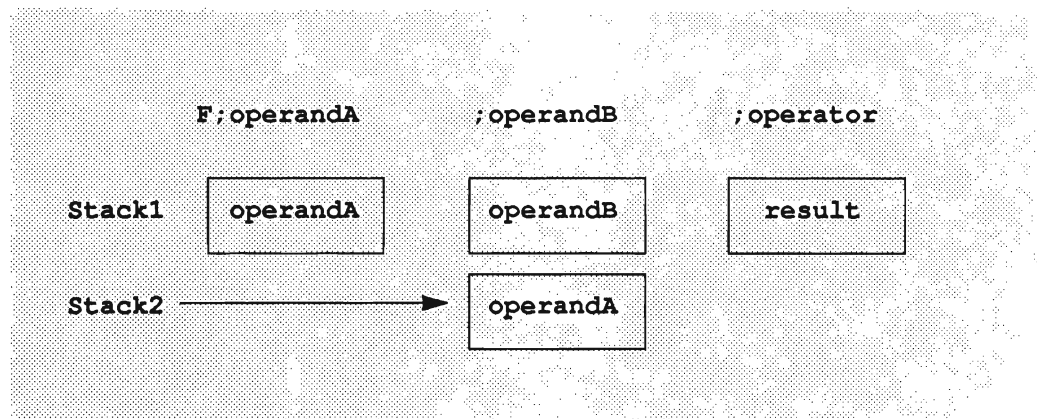
The general form of the F correlative is:

F;parameterA;parameterB;parameterC;...

The command parameters in the F correlative come in two flavors: operands and operators. Operands are data reference parameters which push a new piece of data onto the top of the stack. Operators are commands which perform arithmetic, relational or string functions on the first two entries in the stack to produce a result.

In a generalized form, the stack is handled as follows:

F;operandA;operandB;operator



The simple operands are:

OPERAND	DESCRIPTION
amc	A numeric attribute mark count. Retrieves the data in the referenced attribute. For example, the number 10 retrieves the contents of attribute 10 in the current data item.
Cn	A numeric constant "n." C10 generates the number 10.
D	The system date in internal format.
T	The system time in internal format.
"literal"	A literal constant. This must be enclosed in quotes.

The order of placement of operands in the F correlative is known as Postfix Polish Notation. What this means is that all operations are performed stack2 against stack1. The result is placed in stack 1 and all subsequent stack entries pop up one position.

Stack1 = Stack2 operator Stack1

This standard follows PICK Spectrum (SMA) standard specifications for the F correlative. (Those of you on McDonnell Douglas machines beware; this is all reversed.)

The operators covered in this section perform arithmetic and string concatenation functions.

These operators are:

OPERATOR	DESCRIPTION	
+	Addition	Stack1 = Stack2 + Stack1
-	Subtraction	Stack1 = Stack2 - Stack1
/	Division	Stack1 = Stack2 / Stack1
*	Multiplication	Stack1 = Stack2 * Stack1
:	Concatenation	Stack1 = Stack2 : Stack1

Using the arithmetic operators, write an arithmetic expression to calculate the number of days that payment of the balance is overdue. This is accomplished by determining the difference between the system date (D) and the DATE.DUE (attribute number 9).

system date - date due

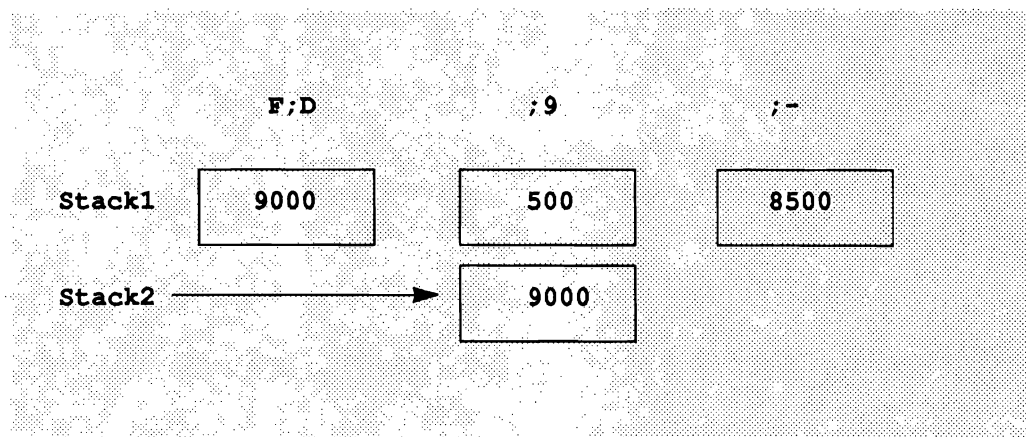
Add the following definition item, DAYS.OVERDUE, to the DICT of CUSTOMERS.

```

DAYS.OVERDUE
001 S
002 0
003
004
005
006
007
008 F;D;9;-
009 R
010 10

```

Using a value of 500 in attribute 9 and an internal system date of 9000, the F correlative (F;D;9;-) is processed as follows:



Enter the following sentence and observe the results:

```
>SORT CUSTOMERS BY DAYS.OVERDUE BALANCE DATE.DUE DAYS.OVERDUE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY DAYS.OVERDUE	sequence clause
BALANCE DATE.DUE DAYS.OVERDUE	output-list

```
>SORT CUSTOMERS BY DAYS.OVERDUE BALANCE DATE.DUE DAYS.OVERDUE<r>
```

PAGE 1

13:53:39 15 DEC 1988

CUSTOMERS. Balance Due. DATE.DUE.... DAYS.OVERDUE

106	126.75	01 MAY 1992	-1233
104	23.42	03 JUN 1991	-900
107	789.32	22 MAY 1991	-888
103	89.12	24 NOV 1989	-344
100	567.80	05 AUG 1989	-233
101	56.75	20 DEC 1988	-5
102	12.34	18 AUG 1966	8155
105	4.56	05 APR 1965	8655

8 ITEMS LISTED.

The results are based on the system date of this writing, December 15, 1988. Of course, the results will differ on your system. Many of the due dates are in the future, making the results negative.

To list only those days that are overdue, enter the following:

```
>SORT CUSTOMERS BY DAYS.OVERDUE...
...WITH DAYS.OVERDUE > "0" BALANCE DAYS.OVERDUE<r>
```

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY DAYS.OVERDUE	sequence clause
WITH DAYS.OVERDUE > "0"	selection clause
BALANCE DAYS.OVERDUE	output-list

```
>SORT CUSTOMERS BY DAYS.OVERDUE WITH DAYS.OVERDUE > "0" BALANCE DAYS.OVERDUE<r>
```

PAGE 1

13:54:17 15 DEC 1988

CUSTOMERS. Balance Due. DAYS.OVERDUE

102	12.34	8155
105	4.56	8655

2 ITEMS LISTED.

8.10.2

Arithmetic Scaling

Since all arithmetic calculations must use integer numbers, numeric values have to be scaled by multiples of 10 to account for implied decimal precision.

For example, interest needs to be charged on the balance due. To make things simple, a straight 10.5 percent is to be added to the BALANCE field.

The general formula is:

$$\text{Interest} = \text{Balance} * (.105)$$
$$\text{Total Balance} = \text{Balance} + \text{Interest}$$

Add the following data definition item to the DICT of CUSTOMERS.

```
ID: NEW.BALANCE
001 S
002 0
003
004
005
006
007 MR25,$
008 F;8;C105;*;8;C1000;*;+
009 R
010 12
```

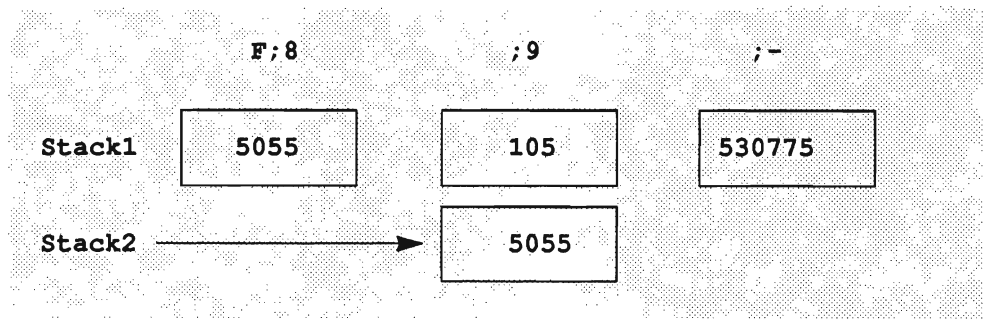
Let's spend some time on this definition item.

First of all, an interest rate of .105 cannot be used in the F correlative. The interest rate must be specified as an implied three decimal number, 105.

Let's say that the BALANCE field (attribute 8) contains 5055. This number is stored with 2 implied decimal places, and therefore, represents 55.55.

The first part of the F correlative to calculate the interest is as follows:

F;8;C105;*...



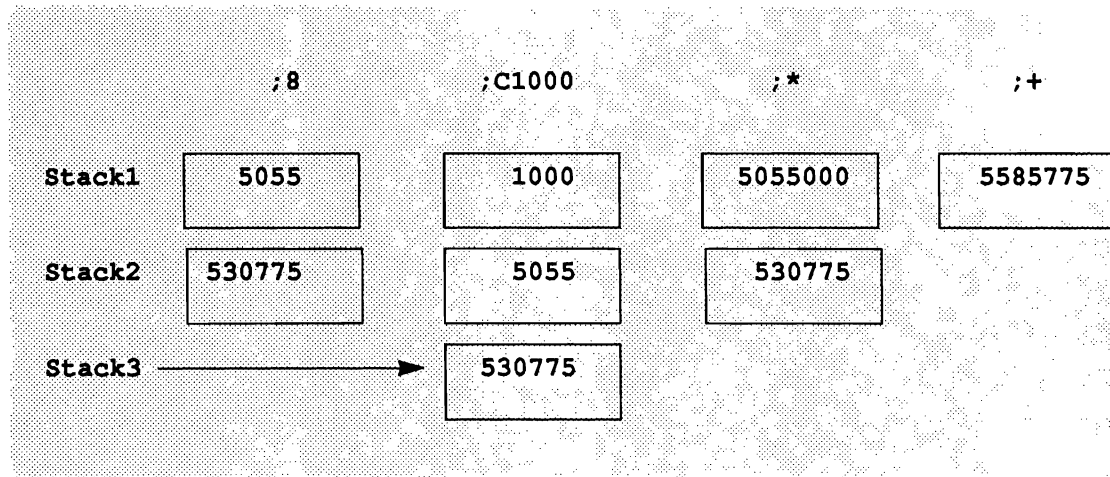
This multiplication of a 2 implied decimal number by a 3 implied decimal number produces a number which has 5 implied decimals (decimal places are added during multiplication). Therefore, the interest charged is actually 5.30775.

To calculate the total balance, the interest must be added to the data value in attribute 8, BALANCE. However, a 5 implied decimal number cannot be added to a 2 implied decimal number without first scaling the 2 decimal number up to 5 implied decimals. This is performed by multiplying attribute 8 by the numeric constant, 1000.

The second part of the F correlative looks like this:

.... ;8;C1000;* ;+

The stack is handled as follows:



The final answer, 5585775, has 5 implied decimal places. The conversion on attribute 007, MR25,\$ says to move the decimal 5 places to the left and round to 2 decimal places.

5585775 → MR25, \$ → \$55.86

Enter the following command sentence and observe the results:

```
>SORT CUSTOMERS WITH DAYS.OVERDUE > "0" BALANCE NEW.BALANCE<r>
PAGE 1                                15:10:24    15 DEC 1988
CUSTOMERS. Balance Due. NEW.BALANCE.
102                12.34          $13.64
105                4.56           $5.04
2 ITEMS LISTED.
```

8.10.3

Concatenation

The F correlative also can be used to build compound strings via concatenation. The following data definition, CSZ, generates the CITY, STATE and ZIP in the same format as is described for the C correlative.

```

      CSZ
001 S
002 0
003
004
005
006
007
008 F;4;" , ";;;5;;; " ";;;6;;
009 T
010 45

```

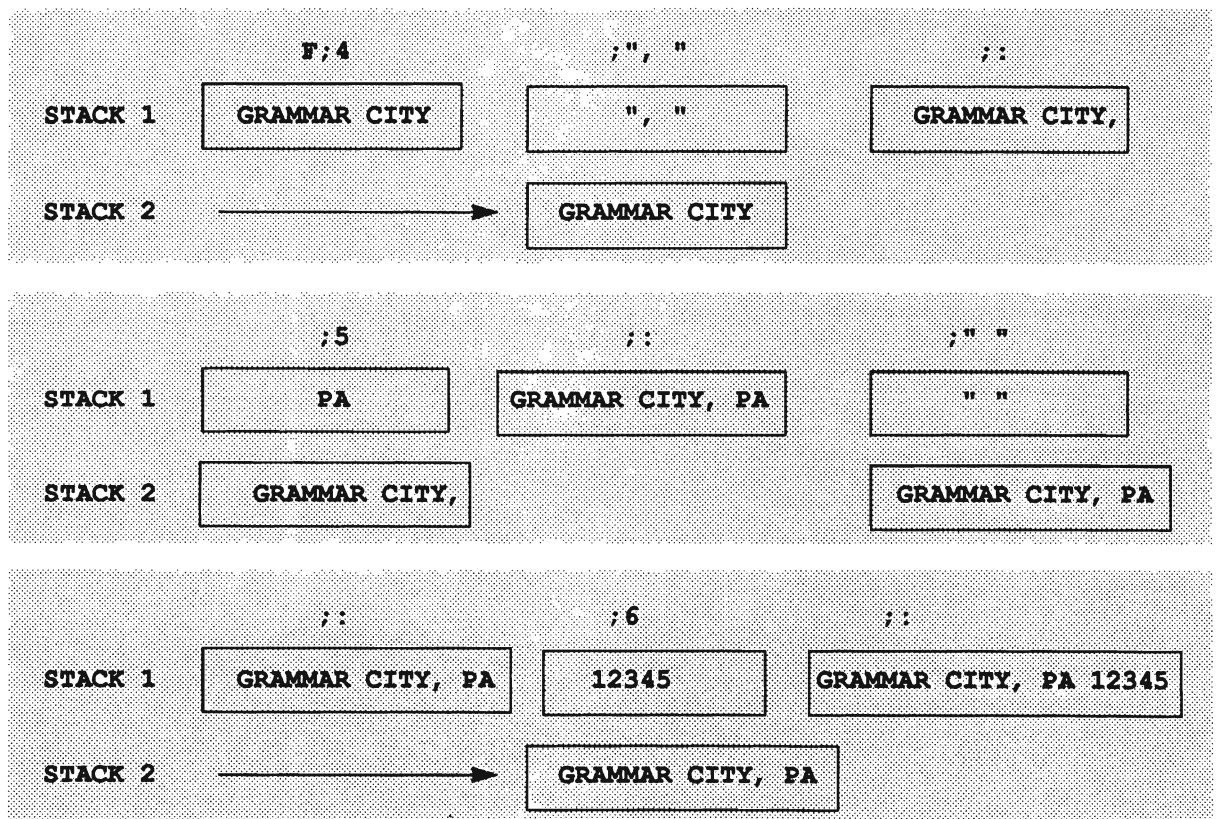
Using item 104:

```

ID: 104
001 SPEAK EASY
002 PARTICIPLE, DANGLY
003 3 DAT DARE ST
004 GRAMMAR CITY
005 PA
006 12345
007 2155554433]2155556321]2155554444
008 2342
009 8555

```

The stack is handled as follows:



Enter the following command sentence and observe the results.

```
>LIST CUSTOMERS CSZ<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
CSZ	output-list

```
>LIST CUSTOMERS CSZ<r>
```

PAGE 1

15:17:09 15 DEC 1988

CUSTOMERS. CSZ.....

106	SANTA ANA, CA 90234
107	VENICE BEACH, CA 97771
100	NEW YORK, NY 10019
101	BROOKLYN, NY 10090
102	WEDONDO BEACH, CA 91234
103	HOLLYWOOD, CA 91329
104	GRAMMAR CITY, PA 12345
105	BYTELAND, CA 90001

8 ITEMS LISTED.

8.10.4

The Simplest F Correlative

The simplest F correlative performs no arithmetic or relational function. It simply generates a constant literal string. For example:

ID: SPACE5

001 S

002 0

003 \

004

005

006

007 F; "

008

009 L

010 5

Suppress the column heading.

Generate a literal 5 spaces.

It doesn't matter if you place the correlative in attributes 007 or 008. This definition always outputs 5 spaces. Enter this into the DICT of CUSTOMERS and then invoke the following command sentence:

```
>LIST CUSTOMERS ZIP SPACE5 BALANCE<r>
```

SENTENCE ELEMENT	DESCRIPTION
LIST	verb
CUSTOMERS	filename
ZIP SPACE5 BALANCE	output-list

```
>LIST CUSTOMERS ZIP SPACE5 BALANCE<r>
```

PAGE 1

10:54:18

16 DEC 1988

CUSTOMERS. ZIP..... Balance Due.

106	90234	126.75
107	97771	789.32
100	10019	567.80
101	10090	56.75
102	91234	12.34
103	91329	89.12
104	12345	23.42
105	90001	4.56

8 ITEMS LISTED.

8.10.5

Other Operands and Operators

These other operands and operators are presented FYI. The following operands are used for specialized work and are not covered in this workbook.

OPERAND	DESCRIPTION
LPV	Load previous value. This is used when an F correlative exists on attributes 008 and 007. The LPV in the F correlative in attribute 007 retrieves the result of the F correlative in attribute 008.
NB	The number of the break level.
ND	The number of detail lines leading up to the break. Since detail lines are not created until print time, this must be specified in attribute 007. See Section 8.12, "Calculations at Break-Lines."
NI	The item count.
NV	The multivalue count.
NS	The subvalue count.
R(R)	Repeat code. Used to repeat an operation on a multivalued attribute. Usually specified in the form: amcR. The optional R is used to repeat subvalues.

The relational operators vary according to implementation. Both possibilities are shown. It is up to you to verify these.

OPERATOR	DESCRIPTION
=	Equal to If Stack2=Stack1 then Stack1=1 else Stack1=0 If Stack1=Stack2 then Stack1=1 else Stack1=0
#	Not equal to If Stack2#Stack1 then Stack1=1 else Stack1=0 If Stack1#Stack2 then Stack1=1 else Stack1=0
>	Greater than If Stack2>Stack1 then Stack1=1 else Stack1=0 If Stack1>Stack2 then Stack1=1 else Stack1=0
<	Less than If Stack2<Stack1 then Stack1=1 else Stack1=0 If Stack1<Stack2 then Stack1=1 else Stack1=0
[	Greater than or equal to If Stack2>=Stack1 then Stack1=1 else Stack1=0 If Stack1>=Stack2 then Stack1=1 else Stack1=0
]	Less than or equal to If Stack2<=Stack1 then Stack1=1 else Stack1=0 If Stack1<=Stack2 then Stack1=1 else Stack1=0

Special Function Operators:

OPERATOR	DESCRIPTION
R	Remainder of the division: Stack2/Stack1
S	Sum of multivalues in Stack1.
P	Push Stack2 onto the top of the Stack.
-	Switch Stack1 and Stack2.
[]	Extract a fixed length string. Stack1 = Stack3[Stack2,Stack1] Stack2 is the starting character position. Stack1 is the number of characters to extract.
(conv)	Any valid conversion enclosed in parentheses. Stack1 = conv(Stack1)

8.11

ALGEBRAIC CORRELATIVE

8.11.1

Introduction to the A Correlative

The algebraic (A) correlative performs all the functions available to the F correlative specified as an algebraic formula, rather than as Post-Fixed Polish stack parameters.

The general format of the A correlative is:

A;algebraic expression

All of the operands and operators are the same as those described for the F correlative.

OK, so why use it? Because it is far easier to read and to subsequently support algebraic formulas.

Is there any difference in speed? Not noticeably. You see, A correlatives are just a simpler way of expressing F correlatives. In fact, the system internally converts A's to F's during the first stage (parse and compile) of ACCESS processing.

For example, take the DAY.OVERDUE F correlative:

F;D;9;-

This can be rewritten in an A correlative as:

A;D-9

Look at a more complex example: the calculation of the NEW.BALANCE based on 10.5% interest. The F correlative used in the NEW.BALANCE definition is:

F;8;C105;*;8;C1000;*;+

This can now be rewritten as follows:

A;(B*"105")+(B*"1000")

Notice that parentheses are used to group the precedence of arithmetic calculations:

(8*"105")	Calculates the interest.
(8*"1000")	Scales the BALANCE to match 5 implied decimals.

These are considered two separate expressions. When they are added,

expression1+expression2
(8*"105")+(8*"1000")

the result is considered a compound expression.

Replace the F correlatives in DAYS.OVERDUE and NEW.BALANCE with the corresponding A correlatives and see if there is any difference. There should be none.

Concatenation can be performed with an A correlative. Compare the city, state and zip F correlatives to the equivalent A correlative.

F;4;",";5;::;" "6;::

versus

A;4;",";5;" "6

Which would you rather read?

8.11.2

The Recursive Operand

There is one extension to the A correlative which is not available to the F correlative. This operand within the expression provides the capability to make recursive references to other data definition items. In simpler terms, instead of directly pointing to the balance attribute using the numeric amc reference 8, the A correlative can use the actual data definition name, BALANCE.

The format of this special operand is:

N(definition name)

The NEW.BALANCE definition can be altered as follows:

```
ID: NEW.BALANCE
001 S
002 0
003
004
005
006
007 MR25,$
008 A: (N(BALANCE) * "105") + (N(BALANCE) * "1000")
009 R
010 12
```

The N(BALANCE) operand retrieves the data value addressed by the BALANCE definition item. In addition, any conversions and correlatives in attribute 008 of the referenced definition item are processed. Therefore, the recursive operand can use definition items which contain any combination of conversions and correlatives in attribute 008, including other A or F correlatives.

NOTE

Recursive operands do not process any conversions or correlatives in attribute 007 of the definition item being referenced.

The CSZ definition can also be rewritten as follows:

```
ID: CSZ
001 S
002 0
003
004
005
006
007
008 N(CITY) : ", " : N(STATE) : " " : N(ZIP)
009 T
010 45
```

8.11.3

Other Special Operands

These are provided FYI, only.

OPERAND	DESCRIPTION
S(expression)	Sums the multivalues referenced by "expression."
R(expressiona,expressionb)	Produces the remainder of the division of expressiona by expressionb.
expression[expressiona,expressionb]	Performs a string extract on the original expression beginning at the position value indicated by expressiona for a length indicated by expressionb.
expression(conv]conv]..)	Any expression can be passed through a series of conversions. See section 8.12 for a demonstration of how this can be used.

8.12

CALCULATIONS AT BREAK LINES

There are just a couple of things to remember when trying to do calculations at the BREAK and TOTAL lines of a report. First:

ALWAYS PUT THE CORRELATIVE ON ATTRIBUTE 007 OF THE DEFINITION ITEM.

BREAK and TOTAL lines are not generated until print time, and are therefore not yet available to line 008.

One more rule: NEVER USE RECURSIVE OPERANDS IN AN A CORRELATIVE!

The next and last correlative formula calculates the average balance due for each state. This is a calculation heavily dependent on the correct ACCESS command sentence being entered. The data definition that you are to enter needs to be tagged on to the following command sentence:

>SORT CUSTOMERS BY STATE BREAK-ON STATE TOTAL BALANCE<r>

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
BREAK-ON STATE TOTAL BALANCE	output-list

Try this:

```
SORT CUSTOMERS BY STATE BREAK-ON STATE TOTAL BALANCE<r>
```

```
PAGE 1
```

```
11:20:06 16 DEC 1988
```

```
CUSTOMERS. STATE Balance Due.
```

102	CA	12.34
103	CA	89.12
105	CA	4.56
106	CA	126.75
107	CA	789.32

```
*** 1022.09
```

100	NY	567.80
101	NY	56.75

```
*** 624.55
```

104	PA	23.42
-----	----	-------

```
*** 23.42
```

```
*** 1670.06
```

```
8 ITEMS LISTED.
```

The average value for each state is based on the number of detail lines output for each state. This value is available at print time by using the ND operand.

Enter the following data definition into the DICT of CUSTOMERS:

```
ID: AVG.BALANCE
001 S
002 0
003 Average]Balance
004
005
006
007 A; (8/ND) (MR2)
008
009 R
010 12
```

The bracket is
a value mark.

The A correlative follows the format that allows any expression to be passed through a conversion.

A;expression(conversion)

The expression divides the value in attribute 8 by the number of detail lines for each BREAK and TOTAL line and the result is then passed through the MR2 conversion.

This same correlative written as an F correlative is as follows:

F;8;ND;/;(MR2)

Now enter the previous command sentence including the request to TOTAL AVG.BALANCE:

>SORT CUSTOMERS BY STATE BREAK-ON STATE TOTAL BALANCE TOTAL AVG.BALANCE<r>

SENTENCE ELEMENT	DESCRIPTION
SORT	verb
CUSTOMERS	filename
BY STATE	sequence clause
BREAK-ON STATE TOTAL BALANCE TOTAL AVG.BALANCE	output-list

>SORT CUSTOMERS BY STATE BREAK-ON STATE TOTAL BALANCE TOTAL AVG.BALANCE<r>

PAGE 1

11:32:44 16 DEC 1988

**CUSTOMERS. STATE Balance Due. Average.....
Balance**

102	CA	12.34	12.34
103	CA	89.12	89.12
105	CA	4.56	4.56
106	CA	126.75	126.75
107	CA	789.32	789.32
	***	1022.09	204.41
100	NY	567.80	567.80
101	NY	56.75	56.75
	***	624.55	312.27
104	PA	23.42	23.42
	***	23.42	23.42
***		1670.06	208.75

8 ITEMS LISTED.

Notice the redundant average balances on the detail lines. These can be removed by using the DET-SUP modifier on the command sentence. DET-SUPP suppresses all of the detail lines. However, what if the detailed BALANCEs need to be shown? This can be taken care of by using any value other than zero as the dummy amc in attribute 002 of the AVG.BALANCE definition item. Try 99.

ID: AVG.BALANCE

001 S

002 99

003 Average]Balance

004

005

006

007 A; (8/ND) (MD2)

008

009 R

010 12

A dummy amc.

The bracket is
a value mark.

Now enter the previous command sentence and observe the output.

>SORT CUSTOMERS BY STATE BREAK-ON STATE TOTAL BALANCE TOTAL AVG.BALANCE<r>

PAGE 1

11:50:01 16 DEC 1988

CUSTOMERS. STATE Balance Due. Average.....
Balance

102	CA	12.34	
103	CA	89.12	
105	CA	4.56	
106	CA	126.75	
107	CA	789.32	
	***	1022.09	204.41
100	NY	567.80	
101	NY	56.75	
	***	624.55	312.27
104	PA	23.42	
	***	23.42	23.42
***		1670.06	208.75

8 ITEMS LISTED.

There is one more problem. Double check the arithmetic at the BREAK and TOTAL lines. Notice that the CA value "204.41" is not quite right. The total, 1022.09, divided by 5 detail lines, gives a result of "204.418". The correctly rounded response should be "204.42".

Here is where scaling becomes necessary to give the correct result. Change the AVG.BALANCE definition item to the following:

ID: AVG. BALANCE

001 S

002 99

003 Average]Balance

004

005

006

007 A; ((8*"100")/ND) (MR24)

008

009 R

010 12

A dummy amc.

The bracket is
a value mark.

The BALANCE in attribute 8 is scaled up an extra 2 decimal places:

$(8*"100")$

This provides more precision to the calculation:

$(8*"100")/ND$

Attribute 8 already has two implied decimals. This produces an integer with 4 implied decimals. The output conversion must also be changed to match the scaling.

$(\text{average calculation})(MR24)$

The decimal is scaled by 4 and rounded to 2.

Now enter the exact same command sentence and observe the correct results.

>SORT CUSTOMERS BY STATE BREAK-ON STATE TOTAL BALANCE TOTAL AVG. BALANCE<r>

PAGE 1

11:50:56 16 DEC 1988

CUSTOMERS. STATE Balance Due. Average.....
Balance

102	CA	12.34	
103	CA	89.12	
105	CA	4.56	
106	CA	126.75	
107	CA	789.32	
	***	1022.09	204.42
100	NY	567.80	
101	NY	56.75	
	***	624.55	312.28
104	PA	23.42	
	***	23.42	23.42
***		1670.06	208.76
8 ITEMS LISTED.			

EXERCISE**8**

1. _____ are used to alter the printed format of the data so that it is easier to read.
2. _____ are formulas which perform arithmetic, logical or string functions.
3. Correlatives create _____ data.
4. The _____ conversion changes internal dates to external format.
5. The "real time" internal day zero is _____.
6. Internal date 6768 is actually July the 5th, 1986. Match the following date conversions:

_____ D2-	a. 1986
_____ D	b. 6
_____ DWA	c. 12
_____ D/	d. 07-12-86
_____ DM	e. 12
_____ DW	f. 86
_____ DY	g. 07
_____ DD	h. SATURDAY
_____ D2Y	i. 1986
_____ DMA	j. JULY

7. Internal time is kept as the number of _____ from _____.
8. Identify the correct time mask:

_____ 14:55:40	a. MTHS
_____ 02:55PM	b. MTS
_____ 14:55	c. MTH
_____ 02:55:40PM	d. MT

9. Using the data string:

77-1/2 SUNSET STRIP

Fill in the corresponding mask character (MC) conversions.

7712

77-1/2 Sunset Strip

77-1/2

SUNSETSTRIP

77-1/2 sunset strip

-/ sunset strip

10. Match the string masking conversion with the output:

Stored Format	Output Format	Conversion
_____ 999999999	999-99-9999	a. MR%5
_____ 234567	***\$2,345.67	b. MR2\$
_____ 55	00055	c. MR4
_____ 123456	1234.56	d. MR24
_____ 123456789	1234	e. ML(#3-#2-#4)
_____ 555	\$0.55	f. ML#4
_____ 123456	12.3456	g. MR2
_____ 123456	12.35	h. MR2,\$*12

11. Fill in the corresponding text extract (T) or group extract (G) correlatives:

Stored Format	Output Format	Conversion
THIS IS A TEST	IS	_____
12/20/52	20/52	_____
2125551212	212	_____
SMITH, JOHN	JOHN	_____

12. The items in the file called INVOICES are identified by a unique invoice number and contain two main attributes, NAME and ZIP as follows:

FILE: INVOICES

ID: 1001	ID: 1002
001 WANSAKRAKA, POLLY	001 TUTIRIBA, DON
002 92713	002 10019

The sales tax charged is stored by zip code in the ZIPS file. Each zip item contains the STATE and the SALES.TAX as follows:

FILE: ZIPS

ID: 92713	ID: 10019
001 CA	001 NY
002 060	002 085

Complete the amc count and file translate portions of the STATE data definition item found in the dictionary of INVOICES.

ID: STATE
001 S
002 _____
003 _____
004 _____
005 _____
006 _____
007 _____
008 _____
009 L
010 2

13. Assuming that the tax rate has three implied decimals, what would be the file translate correlative and output conversion in the data definition item?

ID: TAX.RATE
001 S
002 _____
003 _____
004 _____
005 _____
006 _____
007 _____
008 _____
009 L
010 2

14. Using the following stored string:

Version 1/Test 3/10019

Fill in the results of the conversions. The bracket (]) is a value mark.

Conversion	Results
GO/]MCU	_____
G/3]TZIPS;X;;1]MCT	_____
G2 1]MCN	_____

15. Match the following F correlatives to the corresponding result using the following item.

ID: 23-1

001 2000

002 1000

Result	Correlative
_____ 3000	a. F;2;1;-
_____ 2000000	b. F;1;2;/
_____ 2000	c. F;1;2;-
_____ 10002000	d. F;1;C1000;*
_____ 2	e. F;1;2;+
_____ -1000	f. F;2;1;:

16. Re-write each of the above F correlatives as an equivalent A correlative.

a. _____

b. _____

c. _____

d. _____

e. _____

f. _____

9

STORED ACCESS COMMAND SENTENCES

This chapter is intended to give you the minimal amount of information to enable you to write very simple PROCs to store your ACCESS commands. Command sentences, normally invoked at TCL, can be written in an item as a stored procedure – PROC for short. This allows modifications to be made and output generated without having to re-enter the entire command sentence.

9.1

THE PROC ITEM

All PROC items must have a special PROC indicator, PQ, in the first attribute. (This is not unlike the requirement of an A or S in the first attribute of a data definition item.) The remaining attributes in the item contain the PROC commands.

```
ID: TEST.PROC
001 PQ
002 command
003 command
004 command
...
```

PROC commands are usually single alphabetic characters placed at the beginning of each attribute. For example, the command O stands for output and uses the form:

Otext Output the "text" string to the terminal display.

A PROC cannot be initiated from TCL unless the PROC item is stored in the master dictionary, MD. The PROC item in the MD automatically becomes a verb, and the item-id is the name of the verb. PROC verbs are usually used to transfer control to PROCs stored in other files. The functional PROC items are then stored in a file created specifically to hold them. (Whenever you get a chance, check out PROCLIB, the shared system PROC library. Short PROCs in the MD transfer control to the complete PROCs in PROCLIB. LISTFILES is a good example to peruse on your own.) However, to keep things as simple as possible, the PROCs discussed in this chapter are stored directly in the MD.

Enter the following:

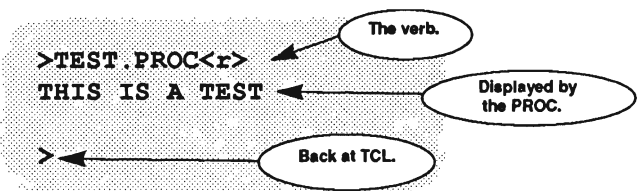
```
>ED MD TEST.PROC<r>
NEW ITEM
TOP
.I<r>
001+PQ<r>
002+0THIS IS A TEST
003+<r>
TOP
.FI<r>
'TEST.PROC' FILED
```

The diagram consists of several callout boxes with arrows pointing to specific parts of the command sequence:

- "EDIT the item TEST.PROC in the master dictionary." points to ">ED MD TEST.PROC<r>".
- "Begin an Insert." points to ".I<r>".
- "PROCS always begin with a PQ." points to "001+PQ<r>".
- "Output the string THIS IS A TEST." points to "002+0THIS IS A TEST".
- "File and exit." points to ".FI<r>".

(This is the last time that I show you the Editor.)

Now invoke the PROC by entering its name at TCL.



9.2 STORING AND PROCESSING COMMAND SENTENCES

9.2.1 *The Primary Output Buffer*

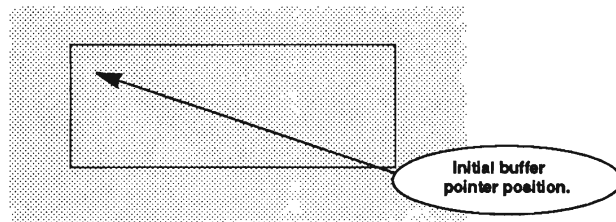
PROC has two internal storage areas called the primary and secondary output buffers. The word "output" is not as descriptive as it could be. For the benefit of the subject at hand, you can think of these as the primary and secondary "command" buffers. ACCESS command sentences are placed in these buffers for subsequent processing.

When a PROC is initiated, both the primary and secondary output buffers are cleared and the primary buffer is active. For simple command sentences, the secondary buffer is not required.

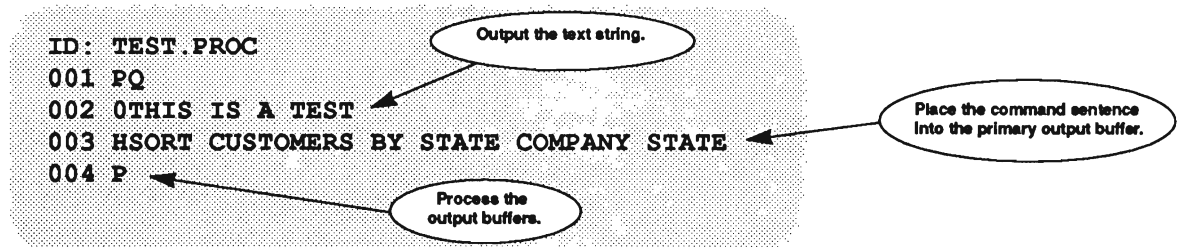
The PROC commands used to fill and process the output buffers are:

COMMAND	DESCRIPTION
Hcommand sentence	Place the command sentence following the "H" into the current buffer pointer position of the currently active output buffer.
P(c)	Process the output buffers. This invokes the command sentences stored in the primary and secondary output buffers. The primary is invoked first and then the secondary. The optional "c" parameter can be any one of the following: - P The PP command means "process and display the output buffers." This helps you see if you've made any mistakes building the output buffers. - W The PW command means "process and wait." The output buffers are displayed, but the PROC waits for a response from the keyboard before continuing. Responses can be: <return> Process the command. - X Exit the PROC. - S Continue the PROC, but skip this command.

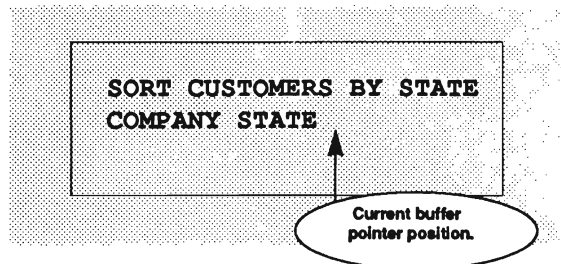
The primary output buffer on PROC initialization:



Modify TEST.PROC in MD so that it looks like this:



Shown below is the primary output buffer resulting from attribute 003 of the above PROC, just prior to processing:



Now, invoke the PROC by entering its name at TCL.

```
>TEST.PROC<r>

THIS IS A TEST

PAGE 1                                06:39:01  17 DEC 1988

CUSTOMERS. Company..... STATE
          Name

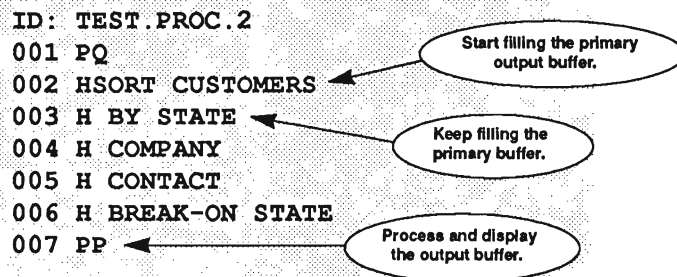
102      THE GUN STORE           CA
103      AEROBIC DEN             CA
105      DIABOLICAL SOFTWARE     CA
106      PHINK, PHINK AND PHINK  CA
107      MONA LISA LTD           CA
100      SHIPS R US              NY
101      ACME CARROT             NY
104      SPEAK EASY              PA

8 ITEMS LISTED.
```

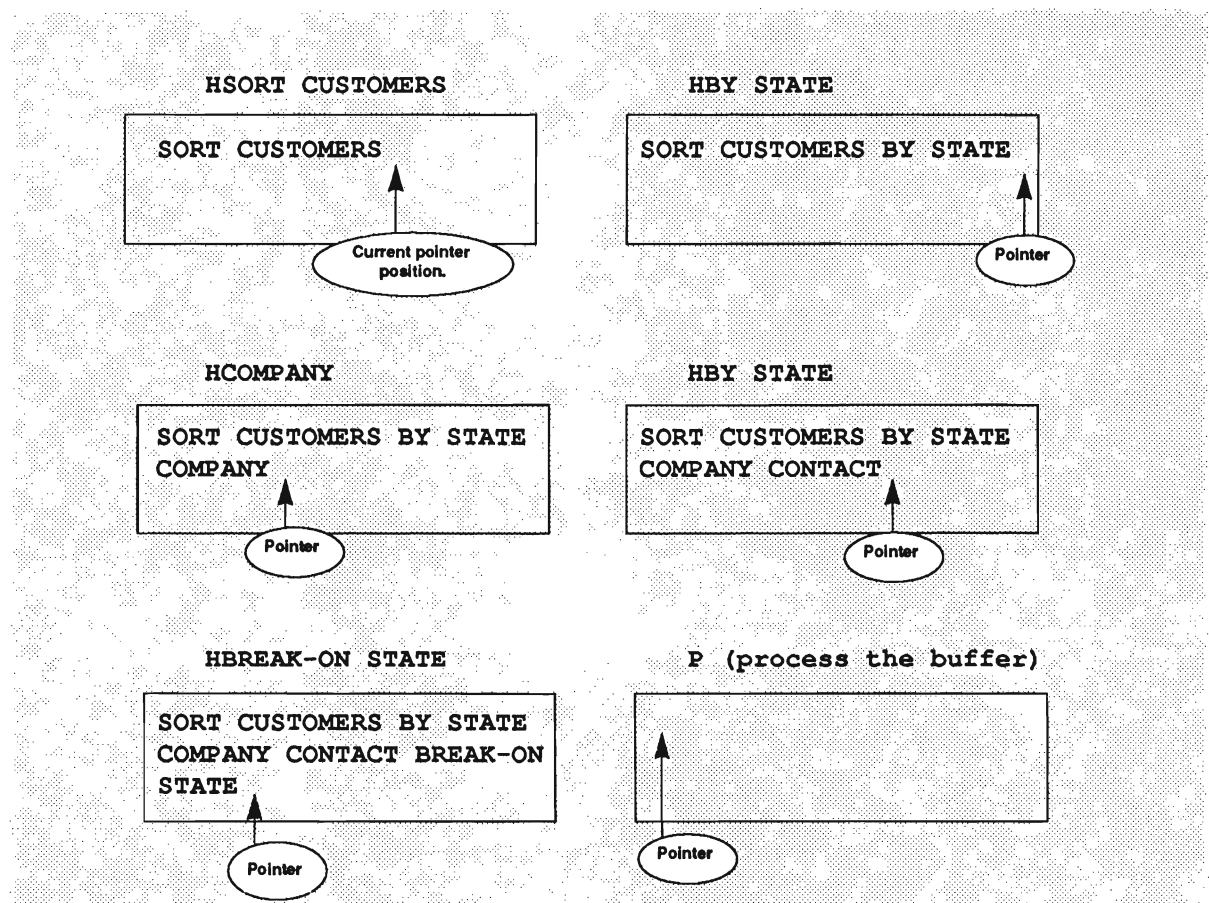
Any ACCESS command sentence can be stored and invoked in this manner.

The entire command sentence need not be placed in a single attribute of the PROC. The PROC output buffer pointer keeps track of the position of the last text string placed in the output buffer. Therefore, multiple H commands keep tagging onto the end of the buffer.

Enter TEST.PROC.2 into the MD:



The primary output buffer as each H command is processed:



Now invoke TEST.PROC.2.

Displayed by the
PP command.

```
>TEST.PROC.2<r>
```

```
SORT CUSTOMERS BY STATE COMPANY CONTACT BREAK-ON STATE
```

```
PAGE 1
```

```
06:45:01 17 DEC 1988
```

```
CUSTOMERS. Company..... CONTACT..... STATE  
            Name
```

```
102      THE GUN STORE      FUDD, ELMER      CA  
103      AEROBIC DEN        PORCELANA, PORCHE CA  
105      DIABOLICAL SOFTWARE GROG, BILL SPIDER CA  
106      PHINK, PHINK AND PHINK PHINK, CLYDE      CA  
107      MONA LISA LTD      DAVINCI, LEONARDO CA
```

```
***
```

```
100      SHIPS R US        HORNBLOWER, HORATIO NY  
101      ACME CARROT      BUNNY, BUGS      NY
```

```
***
```

```
104      SPEAK EASY        PARTICIPLE, DANGLY PA
```

```
***
```

```
***
```

```
8 ITEMS LISTED.
```

9.2.2

The Secondary Output Buffer

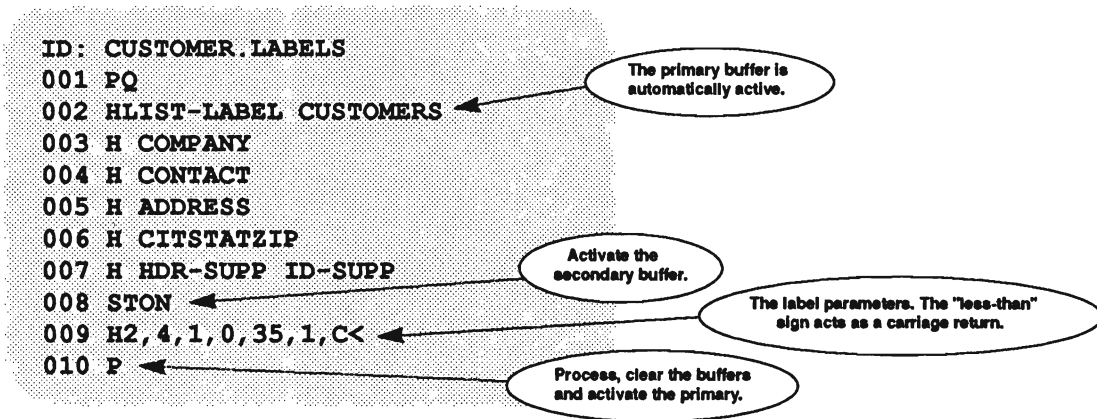
The output buffers are processed in sequence, primary before secondary. The secondary output buffer is also called the "stack" because it is where you stack (store) responses to the questions that will be asked by the process invoked in the primary buffer.

The commands to activate and deactivate the secondary output buffer are:

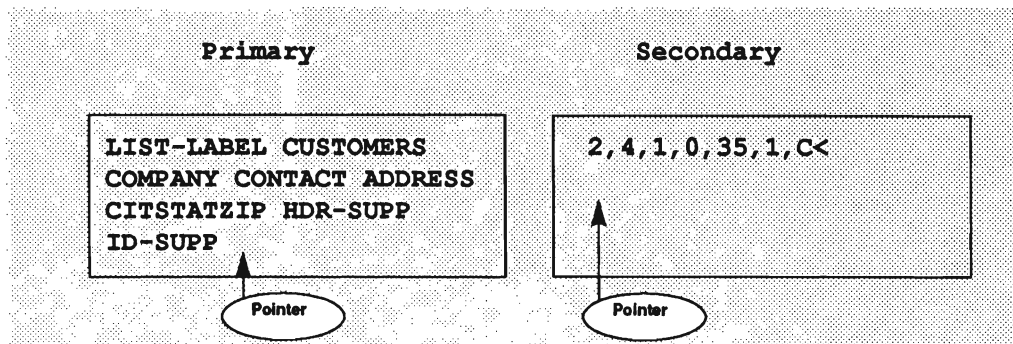
COMMAND	DESCRIPTION
STON	Stack on. Activates the secondary output buffer. All subsequent H commands place the text in the secondary buffer.
STOFF	Stack off. Deactivates the secondary output buffer. This is seldom used since the P command automatically deactivates the secondary.
P	Process the buffers and activate the primary output buffer upon completion.

The STON command is useful when storing mailing label command sentences. The label parameter response can be stored in the secondary output buffer.

Add the following PROC to the MD.



Just prior to processing, the output buffer contents are:



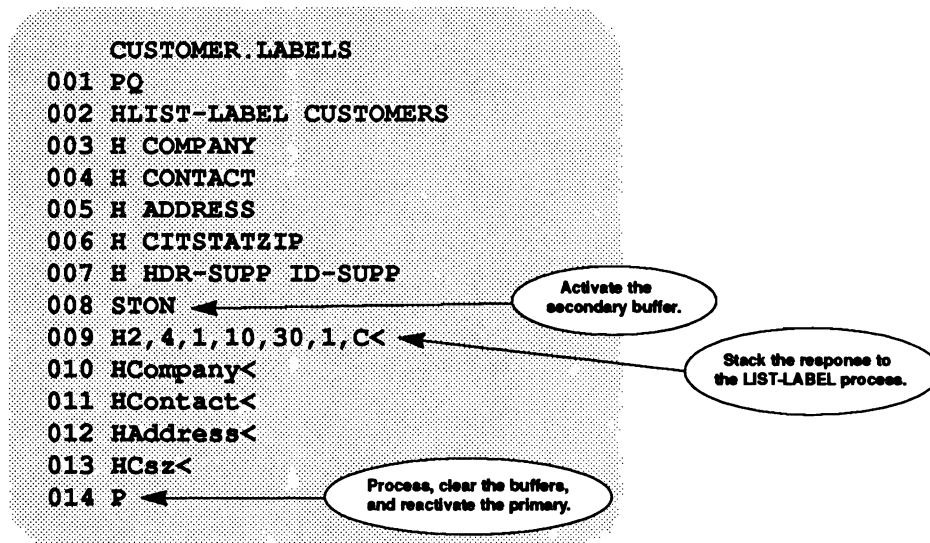
This generates the output:

PHINK, PHINK AND PHINK PHINK, CLYDE 1600 Pennsylvania Ave Santa Ana, Ca 90234	MONA LISA LTD DAVINCI, LEONARDO 15 Century Dr Venice Beach, Ca 97771
SHIPS R US HORNBLOWER, HORATIO 1110-1 W 57Th St New York, Ny 10019	ACME CARROT BUNNY, BUGS 10 Hole-In-Ground Brooklyn, Ny 10090
THE GUN STORE FUDD, ELMER 30 Wabbit Way Wedondo Beach, Ca 91234	AEROBIC DEN PORCELANA, PORCHE 139 Hollywood Blvd Hollywood, Ca 91329
SPEAK EASY PARTICIPLE, DANGLY 3 Dat Dare St Grammar City, Pa 12345	DIABOLICAL SOFTWARE GROG, BILL SPIDER 01100 Binary Way Byteland, Ca 90001

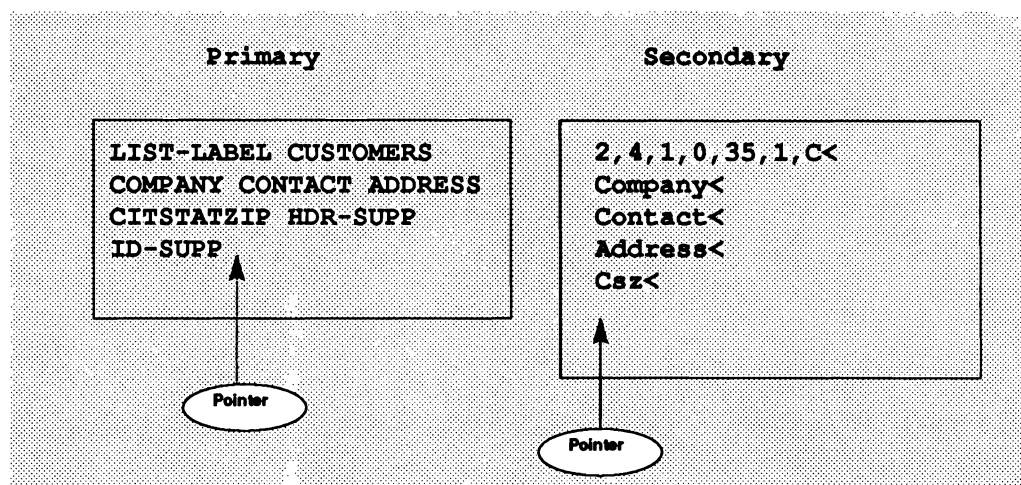
The "less-than" sign (<) at the end of attribute 009 indicates an internal carriage return. Each new entry is "pushed" onto the stack with this indicator. Multiple stack entries answer multiple input requests from the sentence executed in the primary buffer.

For example, if the LIST-LABEL indent parameter is non-zero, the process requires input of label text for each row label. (See Chapter 5, "Mailing Labels.") To perform automatic processing, each response must be pushed on the secondary buffer stack.

Alter the CUSTOMER.LABELS PROC to match the following:



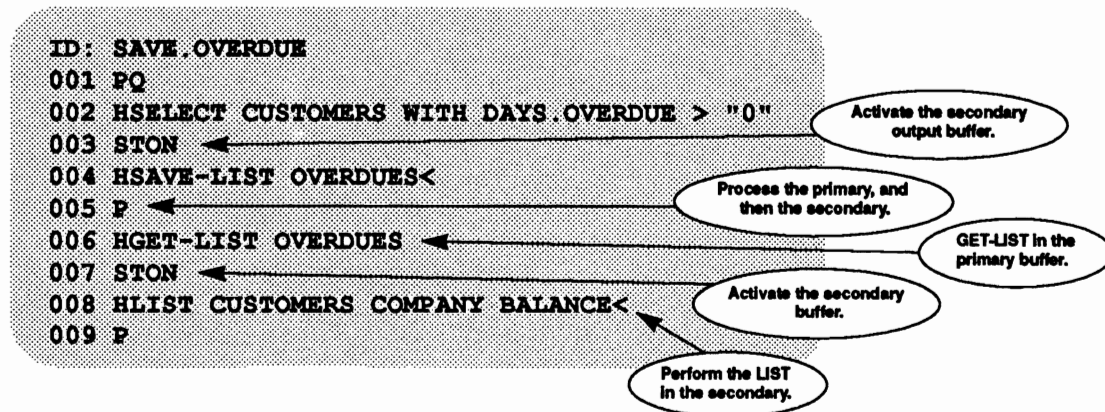
Just prior to processing, the output buffer contents are:



This generates the output:

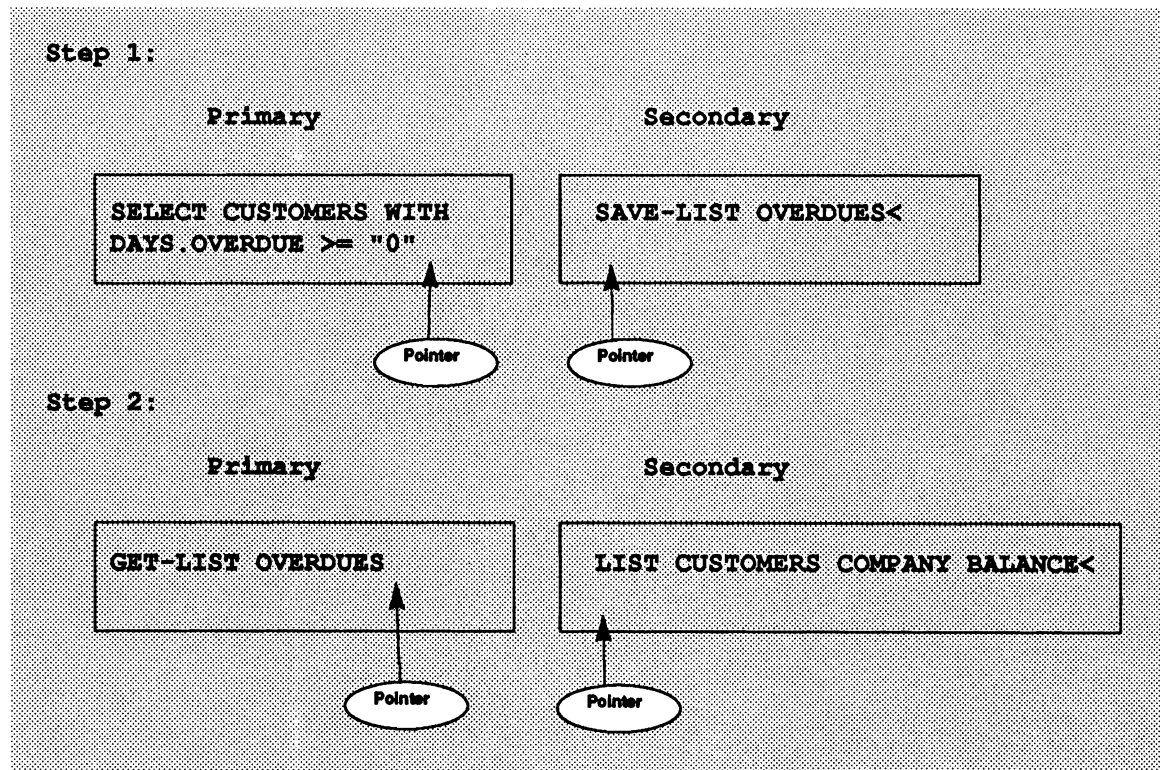
Company	PHINK, PHINK AND PHINK	MONA LISA LTD
Contact	PHINK, CLYDE	DAVINCI, LEONARDO
Address	1600 Pennsylvania Ave	15 Century Dr
Csz	Santa Ana, Ca 90234	Venice Beach, Ca 97771
Company	SHIPS R US	ACME CARROT
Contact	HORNBLOWER, HORATIO	BUNNY, BUGS
Address	1110-1 W 57Th St	10 Hole-In-Ground
Csz	New York, Ny 10019	Brooklyn, Ny 10090
Company	THE GUN STORE	AEROBIC DEN
Contact	FUDD, ELMER	PORCELANA, PORCHE
Address	30 Wabbit Way	139 Hollywood Blvd
Csz	Wedondo Beach, Ca 91234	Hollywood, Ca 91329
Company	SPEAK EASY	DIABOLICAL SOFTWARE
Contact	PARTICIPLE, DANGLY	GROG, BILL SPIDER
Address	3 Dat Dare St	01100 Binary Way
Csz	Grammar City, Pa 12345	Byteland, Ca 90001

The secondary output buffer can also be used to stack other TCL or ACCESS command sentences which use an active list generated by the command in the primary output buffer. For example, add the following PROC to the MD.



This PROC has two main steps. First, it selects and saves the OVERDUES list, and second, it retrieves the OVERDUES list and performs the ACCESS sentence LIST.

Here's what the output buffers look like for each step.



Now, invoke SAVE.OVERDUE.

```
>SAVE.OVERDUE<r>
[404] 2 ITEMS SELECTED. ← System response.
List 'OVERDUES' saved. 1 frame/s used.
[404] 2 ITEMS SELECTED.
PAGE 1                                07:26:36  17 DEC 1988
CUSTOMERS. Company..... Balance Due.
          Name
102      THE GUN STORE           12.34
105      DIABOLICAL SOFTWARE     4.56
2 ITEMS LISTED.
```

EXERCISE**9**

1. The first line of a PROC is always _____.
2. PROCs must be initiated by an item in the _____ file.
3. There are _____ output buffers. The _____ and the _____.
4. The _____ command places text in the currently active output buffer.
5. The command to process the output buffers is _____.
6. Finish the PROC to perform the following ACCESS sentence.

 SORT CUSTOMERS BY STATE BY BALANCE COMPANY STATE BALANCE

 ID: BALANCE.REPORT

001 _____

002 _____

003 _____

004 P

7. The command which activates the secondary buffer is _____.
8. Finish the PROC to perform the following commands in sequence.

 SSELECT CUSTOMERS WITH BALANCE >= "10000.00"

 SAVE-LIST DEAD.BEATS

 ID: SAVE.DEADBEATS

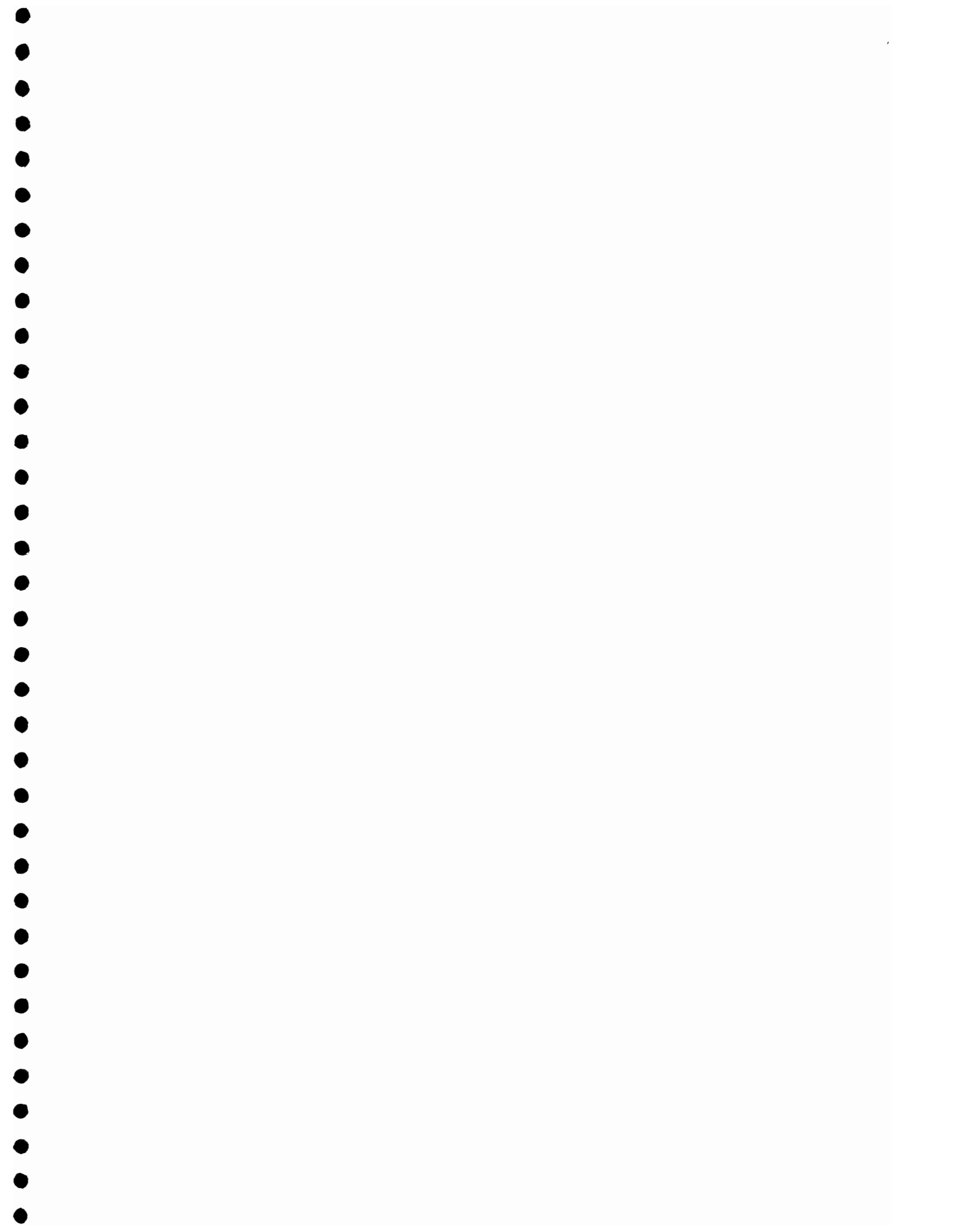
001 _____

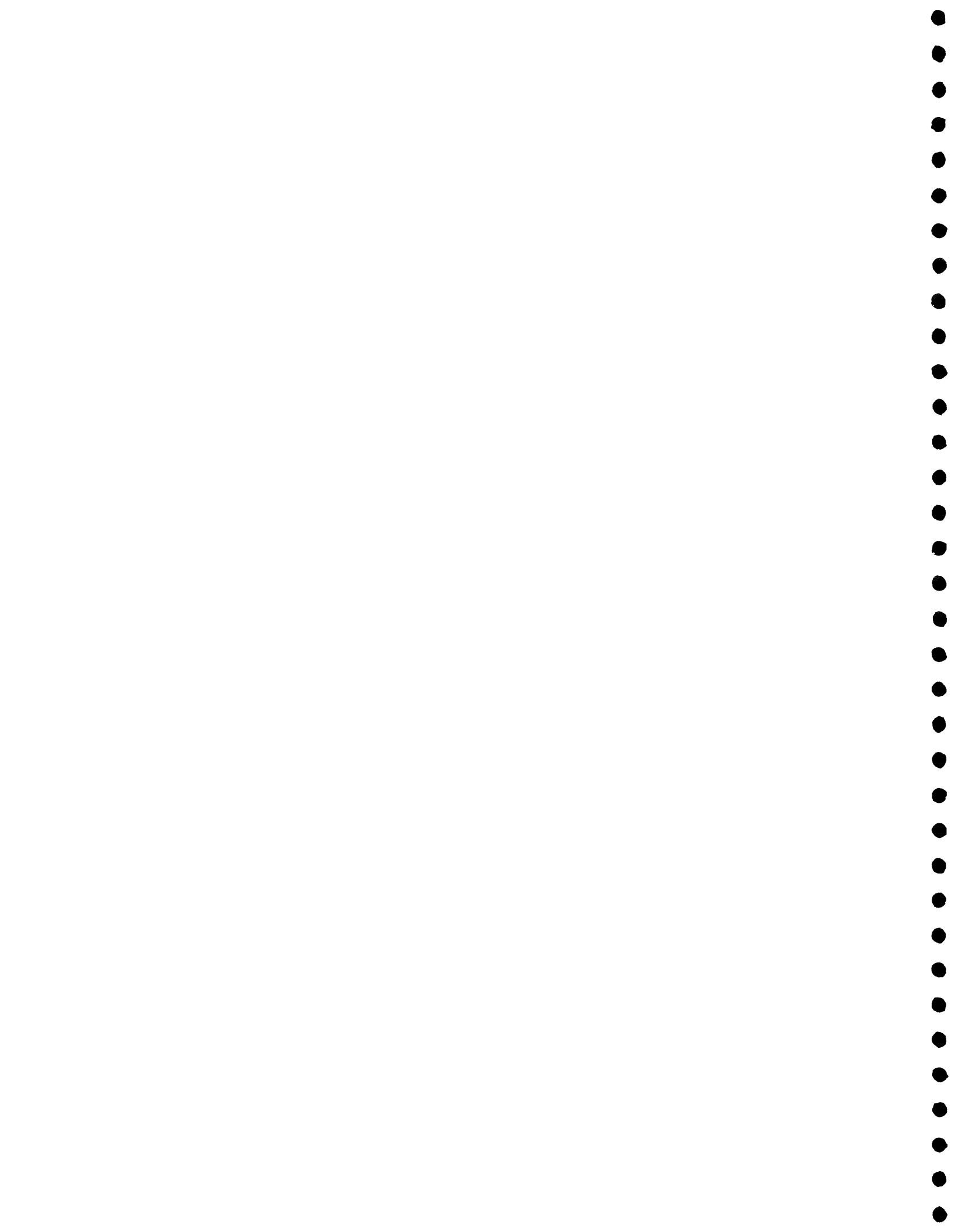
002 _____

003 _____

004 _____

005 P





EXERCISE ANSWERS

EXERCISE 1

1. accounts
2. System Dictionary
3. Master Dictionary
4. items
5. attributes
6. item-id
7. lower (DATA)
8. upper (DICT)
9. attribute mark
10. unlimited
11. 32,000
12. output masking, attribute position, justification, column header, column width
13. false
14. true

EXERCISE 2

1. logging on
2. Terminal Control Language
3. verbs
4. LISTFILES
5. LIST filename
6. SORT filename
7. LIST ONLY DICT CUSTOMERS, or LISTDICT CUSTOMERS, or SORT ONLY DICT CUSTOMERS
8. TCL prompt
9. attribute position pointer

EXERCISE 3

1. verb, filename
2. LIST, SELECT, VERB, COUNT, SORT, LIST-LABEL, SORT-LABEL, SSELECT
3. ordered, random
4. data level
dictionary level
data level
data level
5. COUNT
6. item-ids
7. true
8. LIST CUSTOMERS COMPANY CITY STATE
9. LIST CUSTOMERS '100'101' COMPANY CITY STATE
10. c, d, j, g, f, e, a, b, h, i
11. SORT CUSTOMERS BY CONTACT BY STATE...
...CONTACT STATE
12. g, e, f, b, a, d, c
13. caret (^)
14. LIST CUSTOMERS WITH COMPANY = "[K^" COMPANY
15. SORT CUSTOMERS BY BALANCE...
...WITH STATE = "CA" BALANCE COMPANY STATE, or
SORT CUSTOMERS BY BALANCE...
...WITH STATE "CA" BALANCE COMPANY STATE, or
SORT CUSTOMERS BY BALANCE...
...WITH STATE EQ "CA" BALANCE COMPANY STATE
16. b, d, f, e, a, c

EXERCISE 4

1. LIST
2. SORT
3. 4
4. COMPANY, CONTACT, PHONE, and the ITEM-ID

5. implicit
6. output columns
7. only
8. h, f, j, g, c, a, k, i, b, e, d
9. HEADING "LC" HEADER "LL"
10. "VU"
11. SORT CUSTOMERS BY STATE BREAK-ON...
...STATE TOTAL BALANCE DET-SUPP LPTR, or
SORT CUSTOMERS BY STATE BREAK-ON...
...STATE TOTAL BALANCE (D) (P)
12. TERM 132,65
13. TERM,,,,,,80,55
14. f, c, e, b, g, a, d

EXERCISE 5

1. LIST-LABEL, SORT-LABEL
2. a - columns
b - rows
c - lineskip
d - indent
e - width
f - spacing
g - C (compress)
3. LIST-LABEL CUSTOMERS COMPANY ADDRESS...
...CITY STATE ZIP (IHP)
?3,5,1,0,30,1,C

EXERCISE 6

1. 1st - parses and compiles the sentence
2nd - data sampling, selection, sequencing
3rd - output
2. SELECT and SSELECT
3. false
4. true
5. SELECT CUSTOMERS WITH STATE = "CA" BY COMPANY
6. d, c, a, b, e
7. SELECT CUSTOMERS WITH STATE = "CA" BY BALANCE
SAVE-LIST
8. GET-LIST
LIST CALCUSTS COMPANY ADDRESS STATE
9. COPY-LIST
BAK.CALCUSTS
10. COPY-LIST
(LISTS)
11. DELETE-LIST CALCUSTS
12. DC
13. SORT DICT POINTER-FILE *A5

EXERCISE 7

1. 009
007
002
010
001
008
2. LIST DICT CUSTOMERS WITH D/CODE = "S"
3. <ctl><shift><|>, an ASCII character 253
4. g, d, e, h, a, c, b, f
5. 015, L
6. extend the width of the output column, and text justification
7. \ (backslash) in attribute 003, and 0 (value) in attribute 010

EXERCISE

ANSWERS continued

EXERCISE 8

1. conversions
2. correlatives
3. virtual
4. D (date)
5. December 31, 1967
6. d, c, h, a, g, b, i, e, f, j
7. seconds, midnight
8. b, c, d, a
9. MCN, MCT, MC/A, MCL, MC/N
10. e, h, a, g, f, b, c, d
11. G1,1
G1/2
T1,3
G1,1
12. 2
TZIPS;X;;1
13. 2
MR3
TZIPS;X;;2
14. VERSION 1
Ny
310019
15. e, d, c, f, b, a
16. A;2-1
A;1/2
A;1*2
A;1*"1000"
A;1+2
A;2:1

EXERCISE 9

1. PQ
2. MD
3. 2, primary, secondary
4. H
5. P
6. PQ
HSORT CUSTOMERS BY STATE BY BALANCE
H COMPANY STATE BALANCE
7. STON
8. PQ
H SSELECT CUSTOMERS WITH BALANCE>="10000.00"
STON
H SAVE-LIST DEAD.BEATS<



