

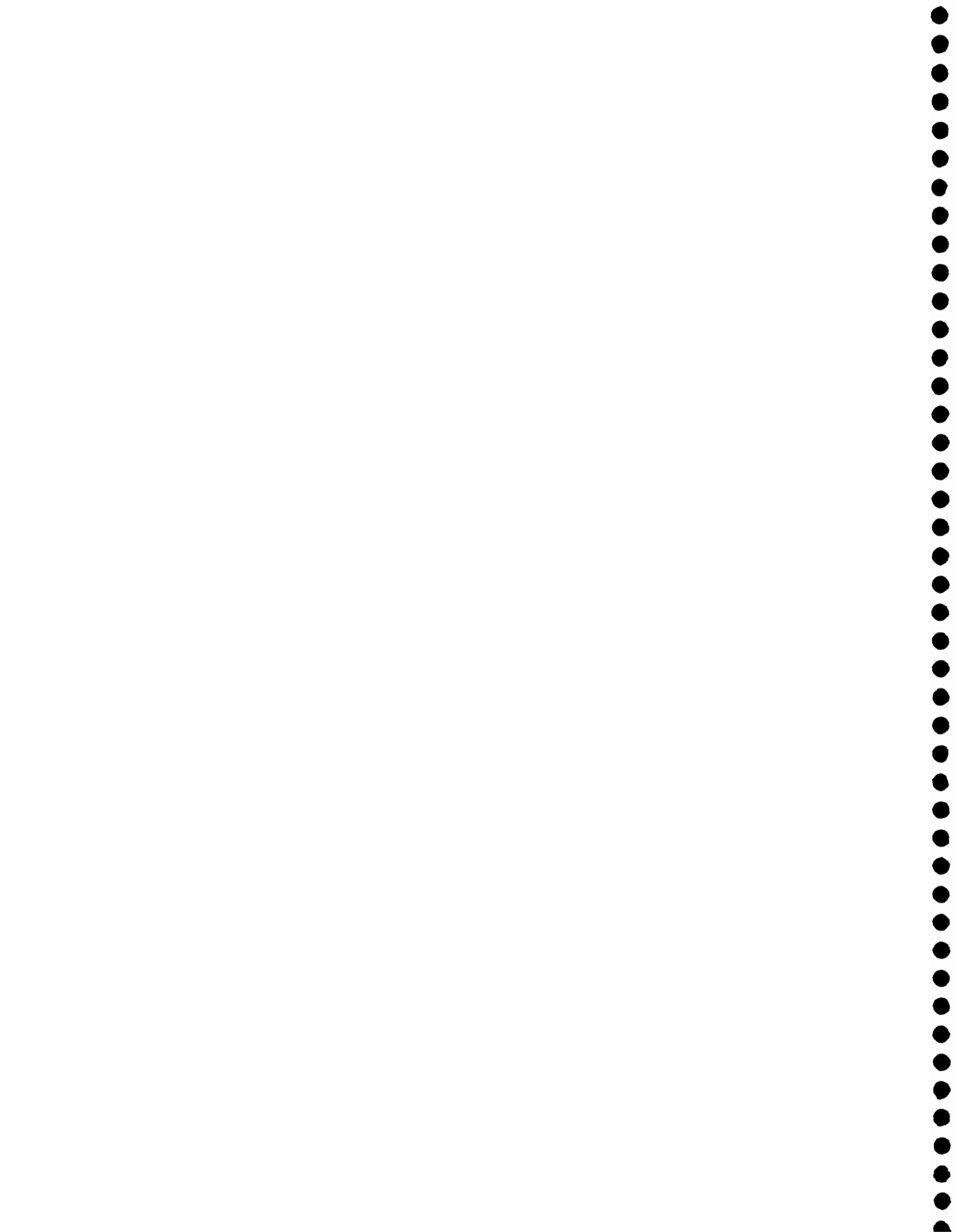
\$19.95

HOW TO

WORKBOOKS FROM IDBMA, INC.

The Editor

by Harvey Eric Rodstein





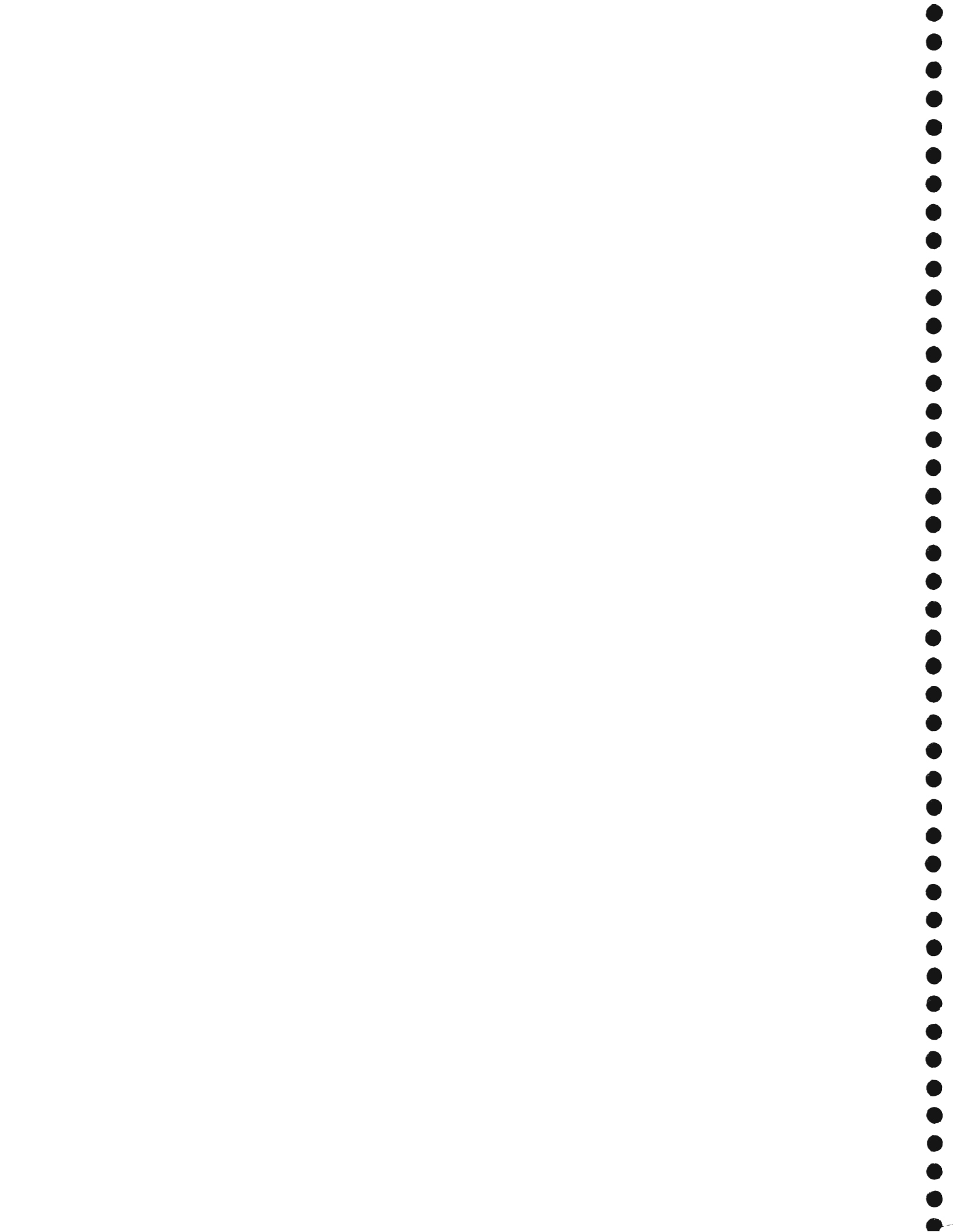
How To
Workbooks From
IDBMA, Inc.

The
E d i t o r

By Harvey Eric Rodstein

© Copyright IDBMA, Inc. 1988
All rights reserved
Printed in the United States

SMA is a trademark of Spectrum Manufacturers Association of San Diego, California. Pick and Pick Operating System are registered trademarks of PICK Systems of Irvine, California.



Dedication

This series of "how-to" books is dedicated to all the people who wanted to use an SMA/Pick-based system but were afraid it was too complicated.

It isn't.

Acknowledgements

Thanks for everything, haneczka, least of which being patience.

Thanks, Leanne, for not shooting me for revision after revision.

Thanks, Gus and Monica, for your trust.

Oh yea, thanks to you for buying this book.

Harvey

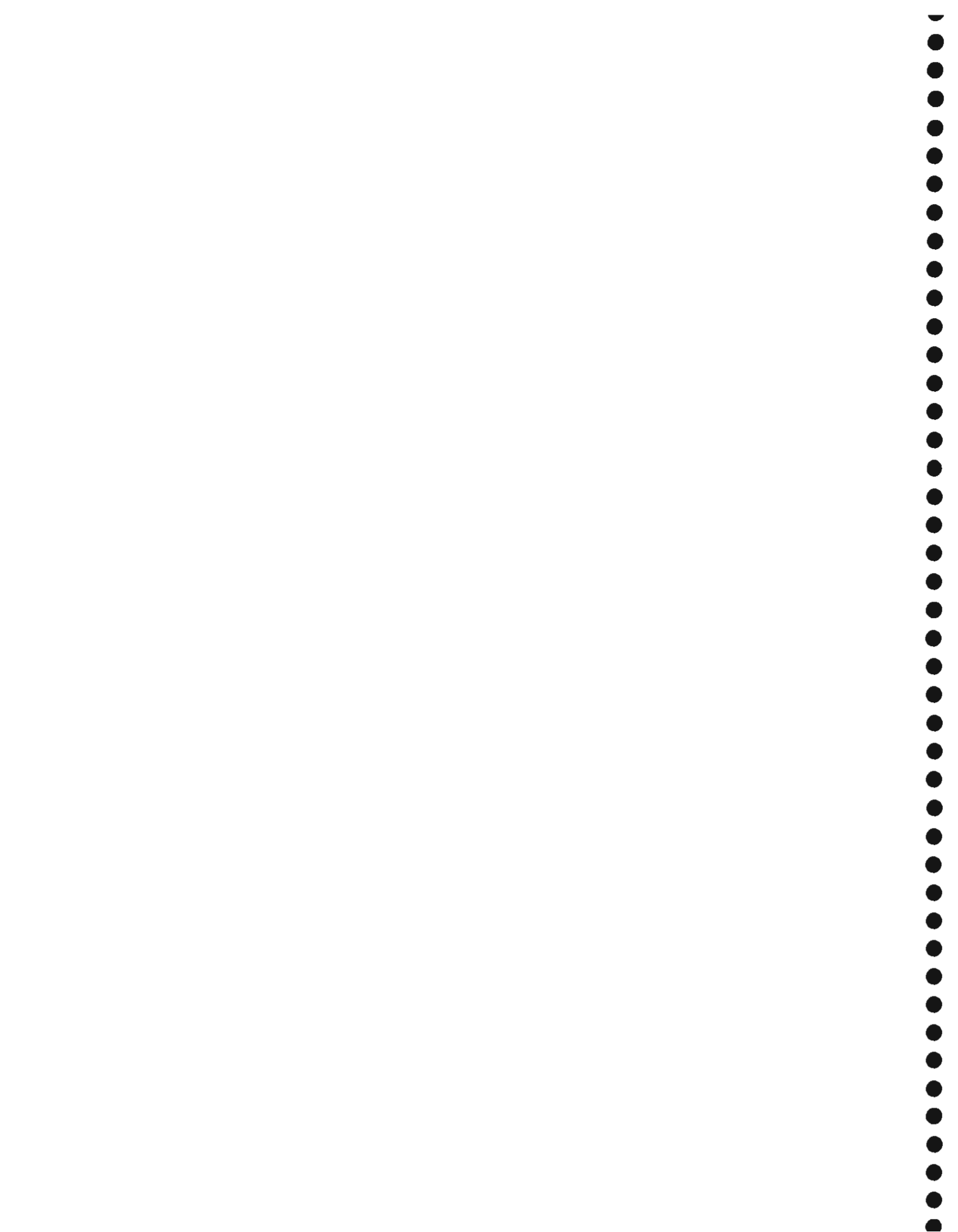


TABLE OF CONTENTS

ABOUT THE EDITOR.	1
1. OVERVIEW	2
1.1 The System	
1.2 Accounts	
1.3 Files	
1.4 Items	
1.4.1 Item-IDs	
1.4.2 Attributes	
1.4.3 Types of Items	
2. GETTING STARTED.	11
2.1 Logging On	
2.2 Looking Around	
2.3 Setting Up Work Files	
2.4 EDIT Verb	
2.4.1 TCL Errors	
2.4.2 Options	
2.5 Editing an Item, New or Existing	
2.5.1 EDIT Command Errors	
2.5.2 Exiting an Item	
2.5.3 Using Item-Lists	
3. GETTING AROUND AN EXISTING ITEM.	23
3.1 Absolute Positioning	
3.2 Relative Positioning	
3.3 Reviewing the Item	
3.4 Getting Item Information	
4. ADDING LINES.	30
4.1 Entering Text from the Keyboard	
4.2 Adding New Lines	
4.3 Adding Lines to an Existing Item	
4.4 Adding Null Lines	
4.4.1 Method 1	
4.4.2 Method 2	
4.5 Setting Tab Stops	
5. SAVING YOUR WORK	49
5.1 The Work Item	
5.2 Saving an Item to the Same Item-ID	
5.3 Saving an Item to a Different Item-ID	
5.4 Exiting Without Saving	

6.	DELETING.	59
6.1	Deleting an Item	
6.2	Deleting Lines	
6.2.1	Unconditional Delete	
6.2.2	Conditional Delete	
6.2.3	Deleting Lines Using Zones	
6.2.4	Zone Errors	
7.	LOCATING.	71
7.1	Locate the First Occurance of a Test String	
7.2	Locate the First Occurance in Multiple Lines	
7.3	Locate Within Zones	
7.4	Wildcard Searches	
8.	REPLACING.	82
8.1	Replacing Lines	
8.2	Replacing Portions of a Line	
8.3	Replacing Using Zones	
8.4	Universal Replace	
8.5	Replacing at the Beginning of a Line (Prefixing)	
8.6	Replacing at the End of a Line (Appending)	
8.6.1	Method 1	
8.6.2	Method 2	
8.7	Truncating a Line	
8.8	Wildcard Replaces	
9.	MERGING.	99
9.1	Merging Lines	
9.2	Merging From the Same File	
9.3	Merging From a Different File	
9.4	Merging Errors	
10.	CANCELLING CHANGES.	106
10.1	The Last Change	
10.2	All Changes Since the Last Flip	
11.	PRESTORED COMMANDS.	112
11.1	Assigning Prestored Commands	
11.2	Displaying and Invoking Prestored Commands	
11.3	Using Item-Lists	
12.	OTHER COMMANDS.	117
12.1	Assembler Format	
12.2	Suppress	
12.3	Macro Expansion	
12.4	Zone Display	
	GLOSSARY.	121
	EXERCISE ANSWERS.	123

About The Editor

EDIT is the line editor provided as a standard system process used to build and revise items within a Pick file. It allows new items to be added and existing items to be altered or deleted. Any item in any file accessible to you is fair game for the editor. EDIT is used to repair data, write RUNOFF documentation and create and maintain both PROC, BASIC or Assembler source items.

With the editor, what you enter is what you get. All keystrokes are accepted. It's important to know the editor and be cautious because a few wrong moves can be devastating to your data and to a good night's sleep.

Course Objectives

The purpose of this workbook is to help the SMA™/Pick® novice master the basic functions of EDIT. It is designed as a self-study tool, consisting of step-by-step explanations of the most frequently used editor functions. These functions and their related commands are presented by example and exercise. At the end of each chapter, there is a short quiz to reinforce the topics covered.

Within each chapter, students are encouraged to follow and enter in, on their system, the examples provided. However, not all examples are hands-on. Please note that those examples in the "rounded-corner" boxes are intended for the student to perform and those in the square boxes are provided for observation only.

This is EDIT in its "generic" form. You may have to consult your manufacturer supplied manuals for implementation differences and extensions.

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 each containing information logically stored by account.

The System Dictionary is a file containing those elements which define valid accounts. Before you as a user can do any work within the system, you must designate to which account you wish to be assigned. This process is called LOGGING ON.

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. 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 in the account called SHIPPING. The single most important account for system operations is called SYSPROG from which all system utilities such as tape backup and system verification are invoked.

Each account has a single Master Dictionary. The Master Dictionary is a file containing those elements defining valid files and commands in the account.

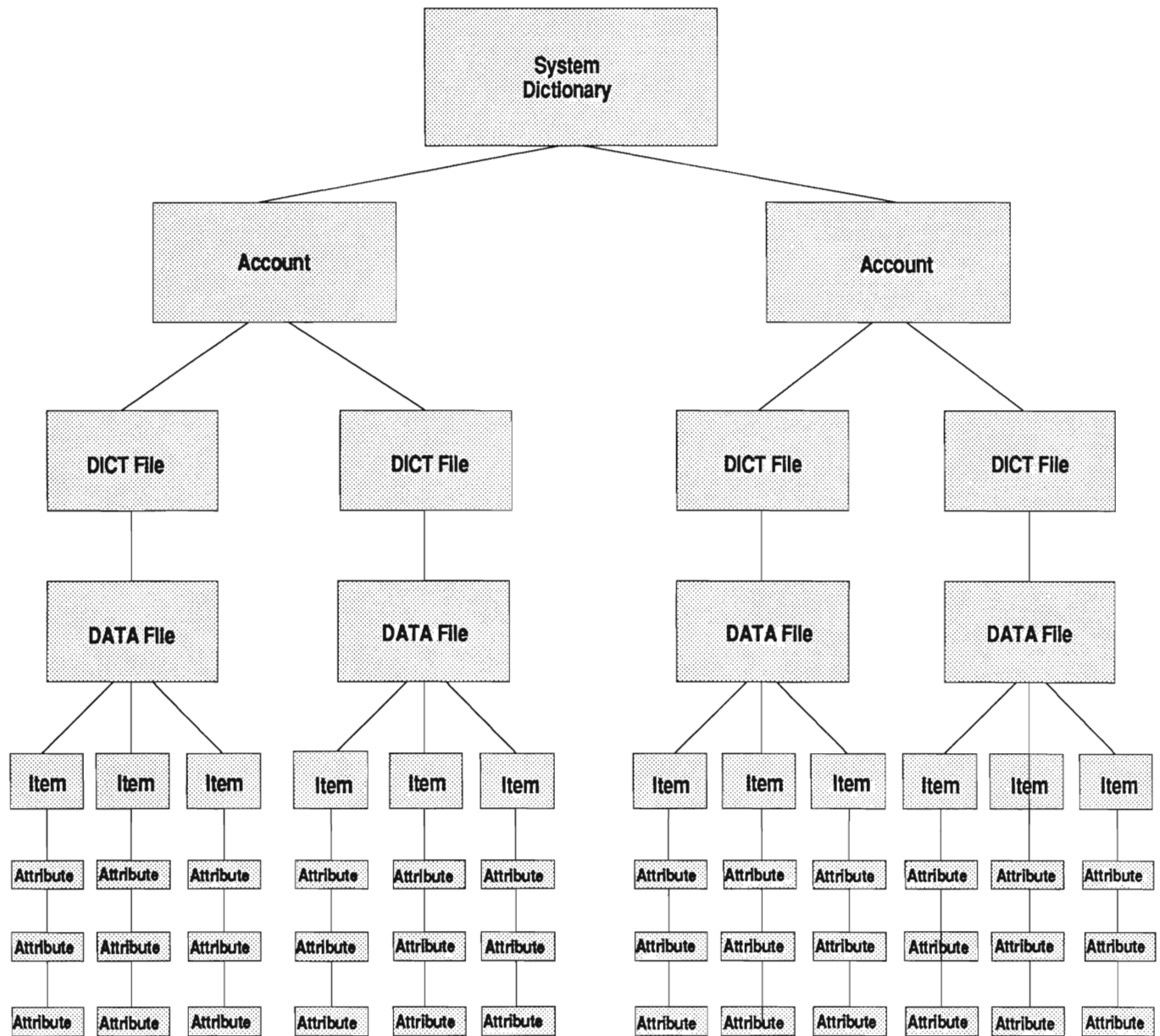
1.3 Files

These are the "cabinet drawers." A file is a set of distinct elements of information grouped according to purpose. Files are usually given names which reflect their use. For example, orders may be kept in a file named ORDERS, while the inventory may be kept in a file called INVENTORY.

Files are actually made up of 2 files or "levels." The "upper" level is called the file DICTIONARY. The "lower" level is called the DATA portion. The DATA portion contains the data and/or text which make up the file's information.

The DICT level contains special items called "Data Definition Items." These are used by ACCESS for retrieving and formatting data. (ACCESS is the Pick Operating System data retrieval language.) EDIT is used to create and modify items in both the DICT and DATA levels of a file.

Hierarchical Diagram of the System



1.4 Items

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

Items themselves are made up of multiple pieces of information. For example, each item in a file named CUSTOMERS represents a single customer. All 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 traditionally called a "field." In this environment they are called "attributes."

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

All data within an item is stored as ASCII characters. (ASCII stands for American Standard Code for Information Interchange).

1.4.1 Item-IDs

Item-ids are used to uniquely identify items within a file and cannot be duplicated. The system automatically passes the item-id through a mathematical algorithm to determine the physical location of the item. This process is known as "hashing" the item. Items in files are stored randomly, not sequentially.

The item-id is a free-form name made up of letters, numbers and punctuations not exceeding 49 characters. This name (IDentifier) is not unlike a reference tab on a file folder. Any retrieval or update of an item must be referenced with both a filename and an itemname (item-id).

1.4.2 Attributes

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

Attributes are variable in length. Please note that in most implementations, a maximum of 32,000 characters is allowed per item.

Sample item100 from the file CUSTOMER as seen in the editor:

ID:	100
001	Caesar, Julius
002	1 Appian Way
003	Olympia
004	WA
005	10101

There are 5 attributes in item 100 above.

The Pick Data Base

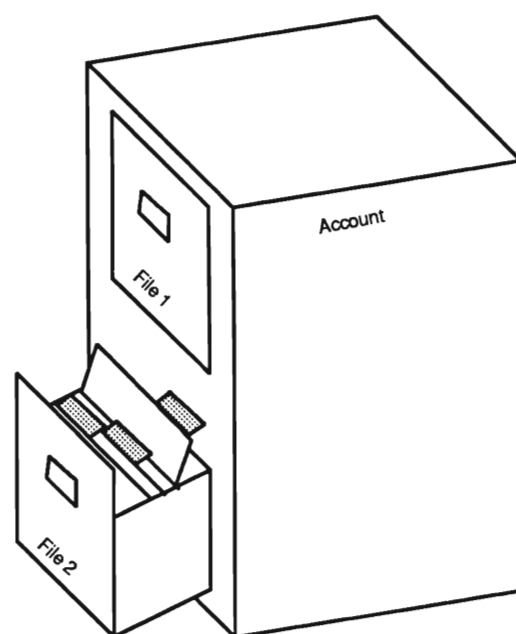
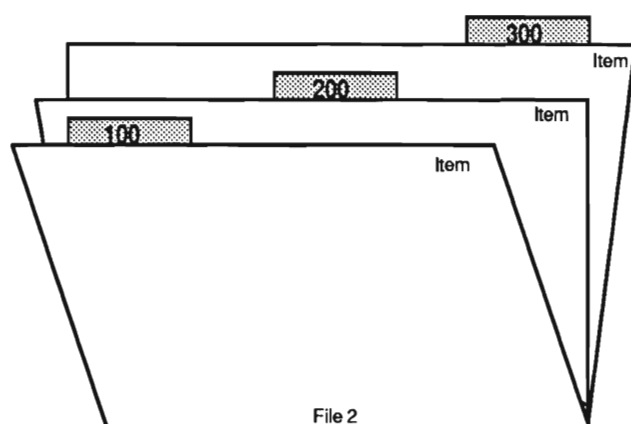
300

Item-ID 300

Attribute: H.E. Rodstein

Attribute: 24332 Toponas Court

Attribute: Laguna Niguel, CA 92656



This is how EDIT represents the item. Each attribute is assigned a line. Referring to line 004 is synonymous to addressing attribute 4.

Attributes are actually marked internally with a special character. This character is used by the system to determine the end of each line and is called an ATTRIBUTE MARK.¹

The physical layout of the above sample item is as follows:

```
100^Caesar, Julius^1 Appian Way^Olympia^WA^10101^_
```

The caret (^) is used to represent an attribute mark. They are displayed by the editor since they indicate the end of a line. The underline character (_) is called a segment mark and is used by the system to mark the end of an item. You don't have to worry about it— it is not part of what you see in the editor.

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

Sample item 123XYZ from the file INVENTORY as seen in the editor:

```
ID: 123XYZ
001 Lawnmower
002 47595
003 101]201]505]303
004 3\10\4]2\5\5
```

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.

The physical layout of the above sample item is as follows:

```
123XYZ^Lawnmower^47595^101]201]505]303^3\10\4]2\5\5^_
```

¹ For those who care, an attribute mark is ASCII character 254. EDIT automatically appends this mark to the end of each entered line.

² A value mark is ASCII character 253 and a sub-value mark is ASCII character 252.

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

Line 3 contains 4 values. The right brackets (]) represent value marks separating each value within attribute 3. The value "303" is the last value in the attribute and does not have to be terminated with another value mark.

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

Line 4 contains 2 values with 3 sub-values each. The backslashes represent sub-value marks separating each sub-value within each value of attribute 4.

NOTE: Do not confuse these special characters with the normal brackets and back-slashes.

1.4.3 Types of ITEMS Most Often Edited

DATA ITEM

A sample DATA Item layout:

```
ID:    Customer Name
001    Name
002    Address
003    City
004    State
```

Would look like this:

```
ID:    100
001    Hornblower, Horatio
002    11110-1 W 57th
003    New York
004    NY
```

```
ID: 101
001 Christian, Spencer
002 1234 Bounty
003 Newport Beach
004 CA
```

TEXT ITEM

ID: NOTES.TO.MYSELF

```
001 Self,  
002 do this,  
003 do that!  
004 don't do that.
```

Each line is free-form text.

PROC ITEM

ID: DAILY.REPORT

```
001 PQ  
002 H SORT THE CUSTOMER FILE  
003 H BY NAME WITH AMOUNT.DUE  
004 A "2  
005 H NAME ADDRESS AMOUNT.DUE  
005 H LPTR  
006 P
```

Line 1 is the PROC indicator "PQ". Always line 1.
Each subsequent line is a PROC statement.

BASIC SOURCE ITEM

ID: SAY.WHAT

```
001 PROGRAM SAY.WHAT  
002 PRINT "SAY.WHAT"  
003 END
```

Each line is a Pick/BASIC statement.

DATA DEFINITION ITEM

Resides in the file DICTIONARY.

ID:	NAME
001	A
002	1
003	Customer Name
004	
005	
006	
007	
008	
009	L
010	35

This DICTIONARY item defines the word NAME to be used in an ACCESS statement to report the first attribute (Attribute "A1") of each DATA item as the Customer's Name. Attributes 009 and 010 specify printing information to the system.

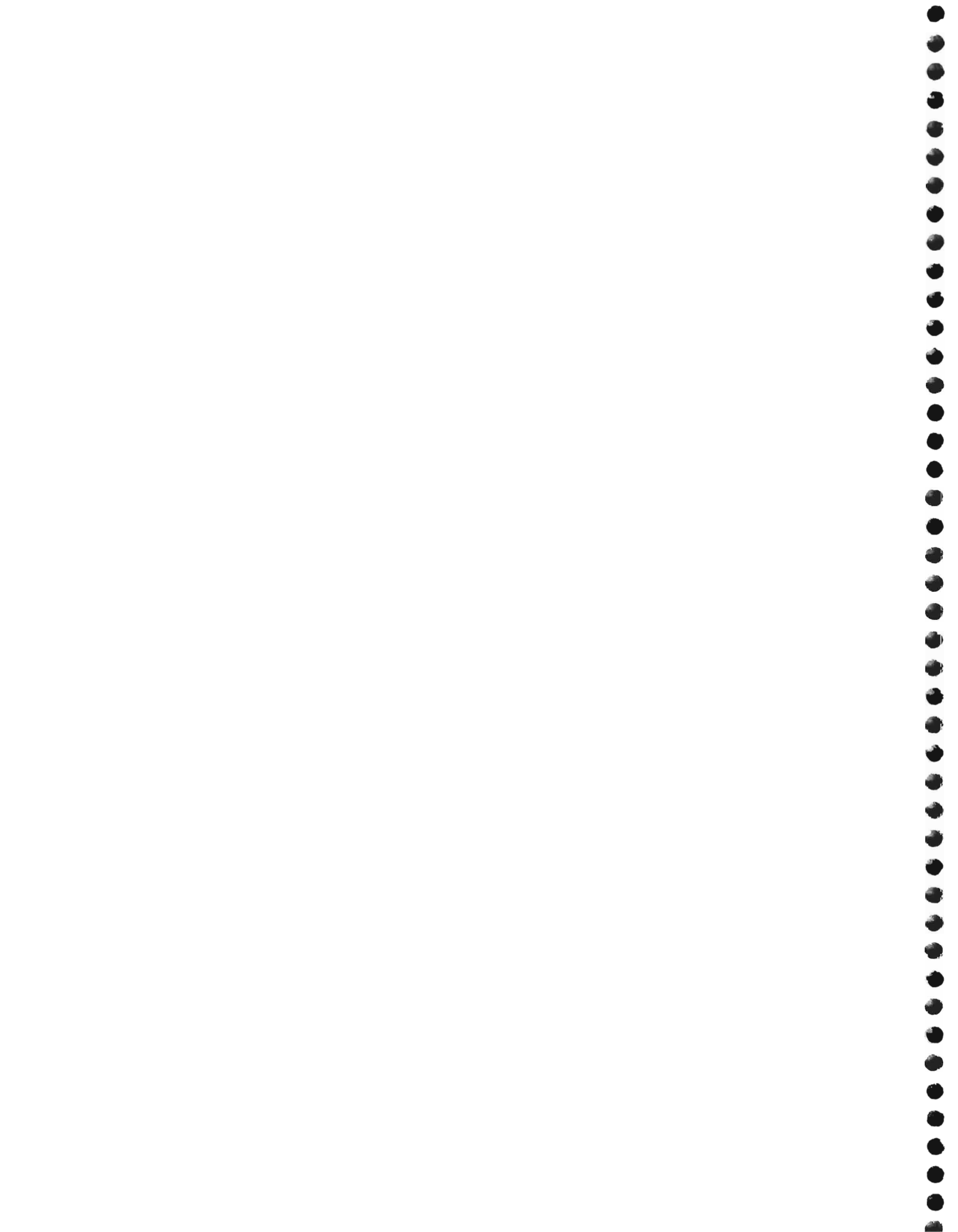
The words data and text are used interchangeably throughout the book since the physical nature of data and text is identical within an item. Everything is stored as ASCII characters.

Exercise 1

1. Files are grouped by _____ .
2. _____ contains the definitions of every account.
3. File and commands are defined in each account's _____ .
4. The _____ contains Data Definition Items.
5. The _____ holds the file's data.
6. Items are retrieved by a unique key called the _____ .
7. Items are made up of lines of information called _____ .
8. Attributes are delimited by a special character called an _____ .
9. Name 3 types of items that can be EDITed.

10. How many attributes can an item hold? _____
11. The maximum number of characters per item is _____ .

NOTES



2. Getting Started

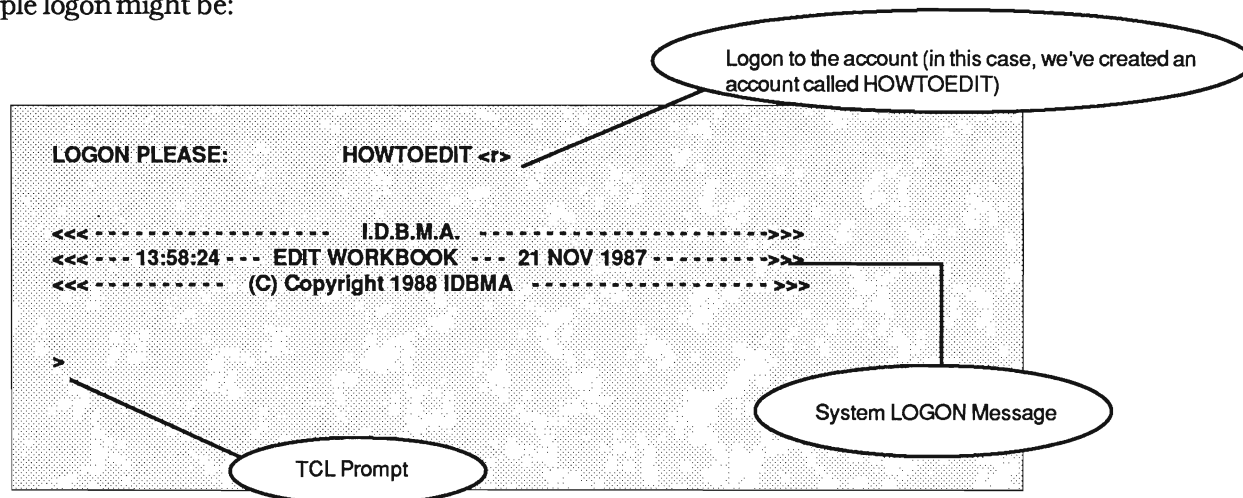
2.1 Logging On

Using an account name to LOGON to the system allows you access to the files and commands (information and services) defined within that account.

If you wish to "cookbook" the examples on a live system and you don't already have an account to logon to, it will be necessary to have one created.

The verb used to create an account is CREATE-ACCOUNT. Accounts can only be created from the SYSPROG account. The details involved in creating a new account are covered in the manuals provided with your system. For the sake of getting to the point, those details are not covered within this text. From this section on, it will be necessary to be logged on to an account in order to perform the examples provided.

A sample logon might be:



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.

Some verbs may "standalone". Such verbs include: TIME, display the current system time and date, WHO, display your port number and account name or LISTU, list the users logged on system. Try these.

Command Sentence:

```
>verb <r>
or
>verb filename item-id <r>
```

PLEASE NOTE: <r> will be used throughout this workbook to indicate <return>

The first word of a command sentence is called a Verb, since it invokes action. (The word "invoke " is used to mean "initiate.") Command sentences used to address file items must include both a file name and an item-id. These verbs include EDIT (that's what we're here for), COPY, to move items within and between files and RUN, to start up a BASIC program.

2.2 Looking Around

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 TCLtype: **>LISTFILES** <r>

Your listing may look similar to this:

Page	1	*** MD *** FILE DEFINING ITEMS		20 FEB 1988
MD.....	CODE	F/BASE	F/MOD	F/SEP
CUSTOMERS	D	6270	1	1
DOCUM	D	5929	1	1
SAMPLES	D	5413	3	1
MEMO	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	
10 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 but defined on another account. The D represents a locally defined file. The Q represents a synonym for a file defined on this or 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 file of your choice. The commands to do this are LIST and SORT. You can try this by using a file name that already exists on your system.

```
>LIST filename <r>
or
>SORT filename <r>
```

LIST generates a listing of the item-ids in random order. In our example, typing at the TCL prompt,

>LIST CUSTOMERS <r>

would give this:

```
PAGE 1                                08:13:39                20 FEB 1988
CUSTOMERS
102
105
100
103
106
101
200
7 ITEMS LISTED.
```

Now try this using a filename that exists on your system.

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

>SORT CUSTOMERS <r>

would give this:

```
PAGE 1                                08:13:39                20 FEB 1988
CUSTOMERS
100
101
102
103
105
106
200
7 ITEMS LISTED.
```

The results of any of the previous commands can be sent to the printer via the system spooler by using the option "P". Options must always begin with a right parenthesis. The left parenthesis is not required.

Try these commands with your own filenames:

>LISTFILES (P) <r>

>LIST Filename (P) <r>

>SORT Filename (P) <r>

2.3 Setting Up Work Files

If you wish to test the examples interactively on your system, it is necessary to create new work files. The files needed are CUSTOMERS, MEMO and SAMPLES.

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 1,1 3,1 <r>
```

```
>CREATE-FILE MEMO 1,1 3,1 <r>
```

```
>CREATE-FILE SAMPLES 1,1 7,1 <r>
```

DICTIONary modulo, separation

DATA modulo, separation

These commands create both the DICTIONary and DATA levels for each file. The parameters used for sizing a file are called a files modulo and separation. A comma separates the two. Modulo specifies the number of statistical groups (sub-files the file will be divided into). The separation determines the number of disk frames that will be contained in a group. Okay, okay, I know what you're saying. What's a frame? WHAT IS A GROUP? The theories and arguments about their use could fill a Techie Review. So with all due regrets, we must move on. You don't have to be acquainted with them to use EDIT.

2.4 The EDIT Verb

The Editor is invoked at TCL by either of the synonymous verbs "ED" or "EDIT." You must specify both the item's filename and item-id in the command sentence.

```
>EDIT filename item-id (options) <r>
```

```
or
```

```
>ED filename item-id (options) <r>
```

```
or
```

```
>EDIT DICT filename item-id (options) <r>
```

```
or
```

```
>ED DICT filename item-id (options) <r>
```

Notice that when invoking EDIT to modify items within the DICTIONary of a file, the reserved word DICT must be used. However, when editing data level items, the DATA designator is implied.

You don't need to say,

```
>ED DATA filename item-id
```

2.4.1 TCL Errors

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

[3] VERB?

If the filename you are requesting to edit does not exist, the following error is displayed:

[201] "filename" IS NOT A FILE NAME.

In either case, check the spelling of the offending entry in the EDIT statement. Notice that a number in brackets often appears before the actual system message.

2.4.2 Options

Options are placed at the end of a command sentence and must be proceeded with a left parenthesis. The right parenthesis is not required. EDIT options are provided here for your information only. In most cases, you will not use them.

<u>Option</u>	<u>Description</u>
A	Assembler formatting. The same as the "AS" EDIT command. Used only if EDIT is being used to enter Assemblersourceitems. (see "OtherCommands", Page ##)
S	Suppress line numbering. Also used to suppress assemblersourceitems. (see "OtherCommands", Page ##)
M	Macro Expansion toggled on and off. The same as the "M" EDIT command. Used only if EDIT is being used to enter Assembler source items. (see "OtherCommands", Page ##)
P	Send all EDIT messages to the print spooler
Z	Suppresses both the "TOP" and the "EOI" (end of item) messages.

Example:

```
>EDIT CUSTOMER 101 (P,S)
```

This edits item 101 in the CUSTOMER file. Suppresses the output of the line numbers and directs all the output to the printer. Notice that there is no TOP message. This combination of options isn't a good idea.

2.5 Editing An Item

To EDIT item 100 in the file CUSTOMERS, the TCL command looks like this:

```
>ED CUSTOMERS 100 <r>
```

Your screen displays:

```
>ED CUSTOMERS 100  
NEW ITEM  
TOP  
.
```

The Editor retrieves this item from the file and places its internal "pointer" at the "TOP" of the item.

The TOP of an item is the position just before the first line.

Since the item does not already exist, EDIT displays the message,

```
"NEW ITEM"
```

and places you at the "TOP" of the item.

If an item already exists, then the TOP message is the only message that appears.

```
>ED CUSTOMERS 500 <r>  
TOP  
.
```

Command prompt.

2.6 Edit Operational Modes

The editor has two modes of operation: the command mode and the text (or data entry) mode.

The command mode is signified by a period (.) and is active by default when an item is edited. The period is a prompt similar to the greater-than (>) prompt used in TCL. The period signifies that the editor is waiting for a command to be entered from your keyboard. Editor commands allow movement within the item, review of any portion of the text within the item and modification of text. While command mode is active, keystrokes are accepted as commands and not as text.

The text entry mode is signified by the plus sign (+), text entry is initiated by either the (I) insert or replace commands. This mode accepts keystrokes as data to be added to the item.

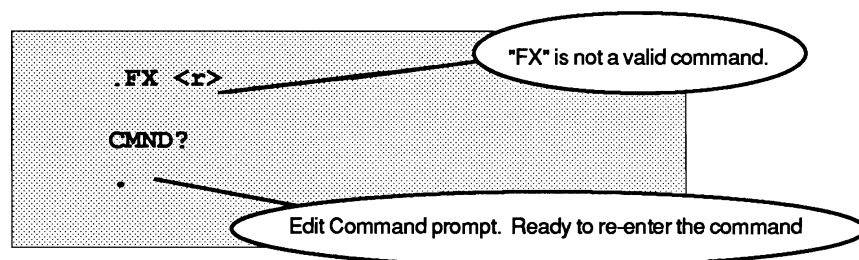
2.5.1 Edit Command Error

If an EDIT command is invalid, the message,

CMND?

appears. (System messages occurring within EDIT do not have preceding numbers.)

The command must be re-entered correctly:



2.5.2 Exiting Edit

There are two ways to leave the EDIT process.

To EXIT the item without saving the results, use:

<u>Command</u>	<u>Description</u>
EX	EXIT the editor without saving the item. Automatically EDIT the next item if an item-list is active. (See below.)
EXK	EXIT the editor without saving the item and return to TCL. The "K" means "kill" an active item-list.

Exiting (EX) an item without saving it is useful to review data non-destructively, or back out of any changes made since your last save command. Try this:

>ED CUSTOMERS 100 <r>
NEW ITEM
TOP
.EX <r>
'100' EXITED
>

EXIT is confirmed

Exiting a single item returns you to the TCL prompt. If an item-list (see next section) is involved, the EDIT process is automatically invoked on the next item.

TO EXIT the item and SAVE the results, use:

Command	Description
FI	FILE (SAVE) the item and exit EDIT. If an item-list is active, EDIT the next item.
FIK	FILE the item and exit EDIT. The K means "kill" any active item-list and return to TCL.

Try this:

>ED CUSTOMERS 100 <r>
NEW ITEM
TOP
.FI <r>
'100' FILED
>

FILE and EXIT

FILE and EXIT is confirmed

We've just filed a null (empty) item. Don't worry. We'll learn how to delete it later.

2.5.3 Using Item-Lists

EDIT can also be invoked for more than one item. This specification is called an item-list.

```
>ED filename item-list
or
>ED DICT filename item-list
```

An item-list can be a sequence of item-ids separated by spaces,

```
>ED filename item1 item2 item3 item4
```

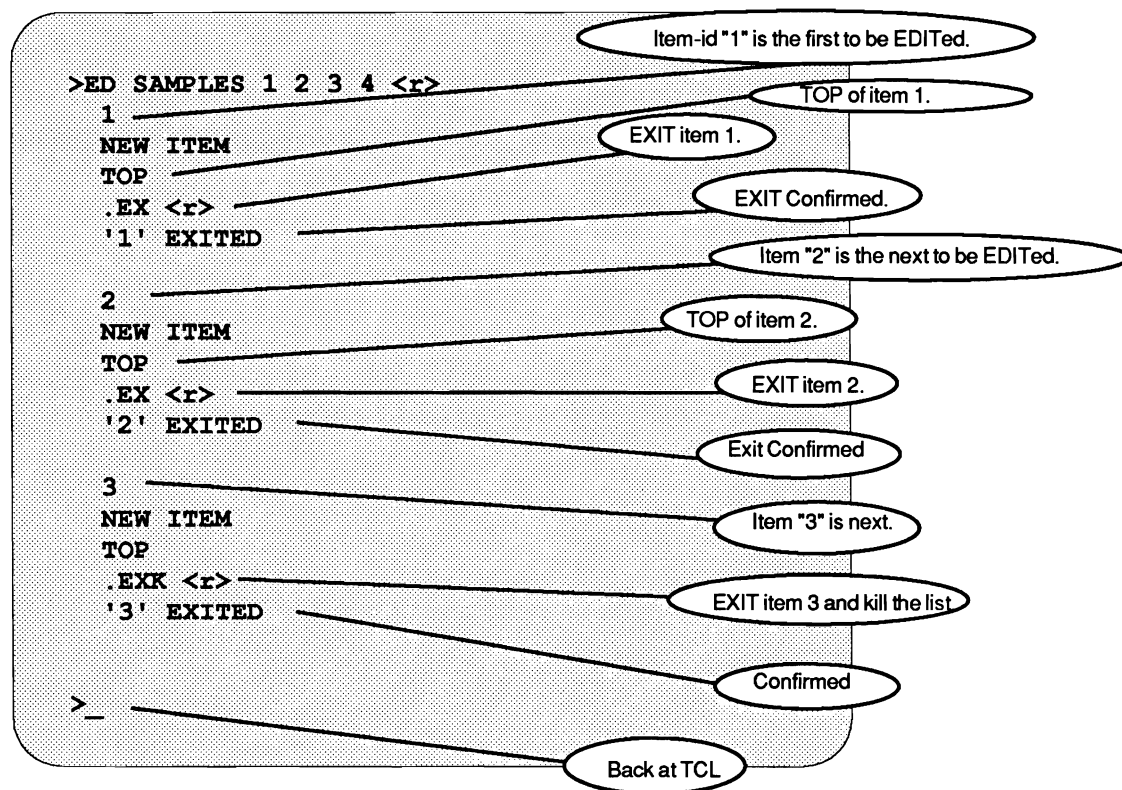
which will EDIT each of these 4 items in sequence, or by an asterisk (*),

```
>ED filename *
```

to EDIT every item on file.

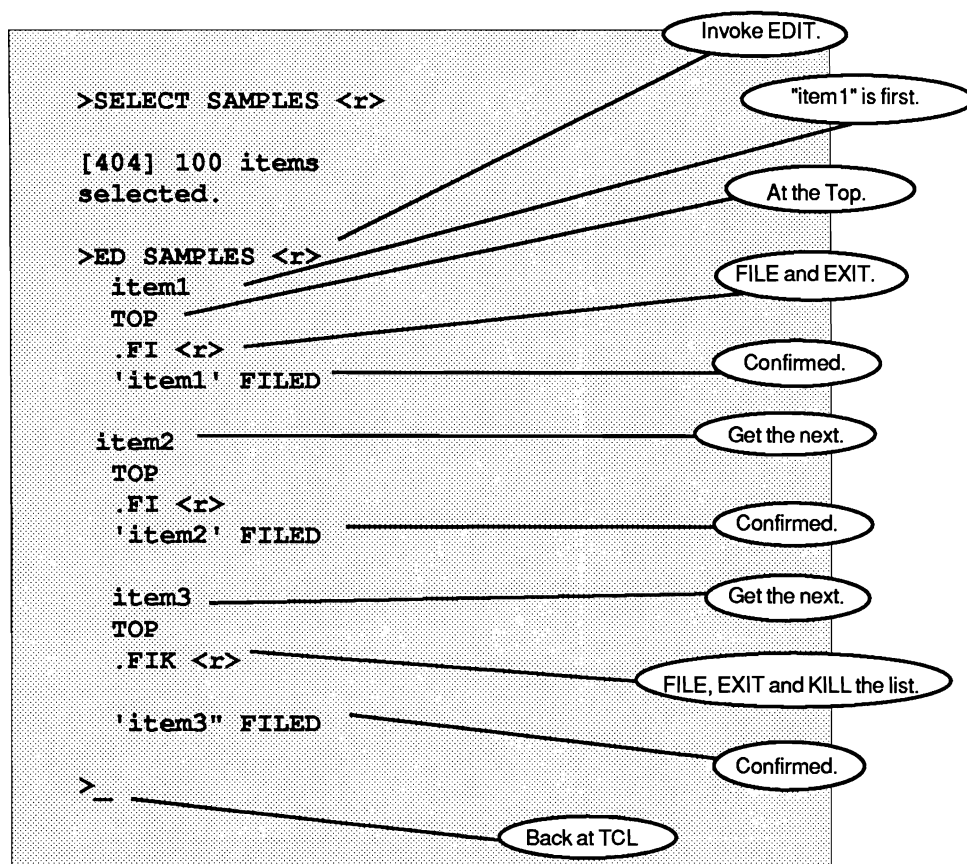
Example,

EDIT 4 items (1,2,3 and 4) in the file SAMPLES.



If items already exist on a file, an item-list can be built using an ACCESS statement using either the verb SELECT, SSELECT, GET-LIST, or QSELECT. Such an item-list is referred to as a "select-list." If a select-list is active, then the item-id can be omitted from the EDIT statement. Each item on the ACCESS list will be EDITed in turn.

Example:



The section entitled "Sample Items" contains the items used as examples throughout this book. These items can be entered as an exercise in the chapter entitled "ADDING LINES".

Exercise 2

1. Files under an account are made available for use by entering the account name at _____ .
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 all item-ids within a file.
6. _____ produces a sorted listing of item-ids within a file.
7. Enter the command to EDIT item A20 in the INVENTORY file.
> _____
8. A20 does not already exist. Enter the system response.

.
.
9. What does the period (.) represent? _____
10. Fill in the command that is invoked.
.

'A20' EXITED
>
11. What does the ">" represent? _____

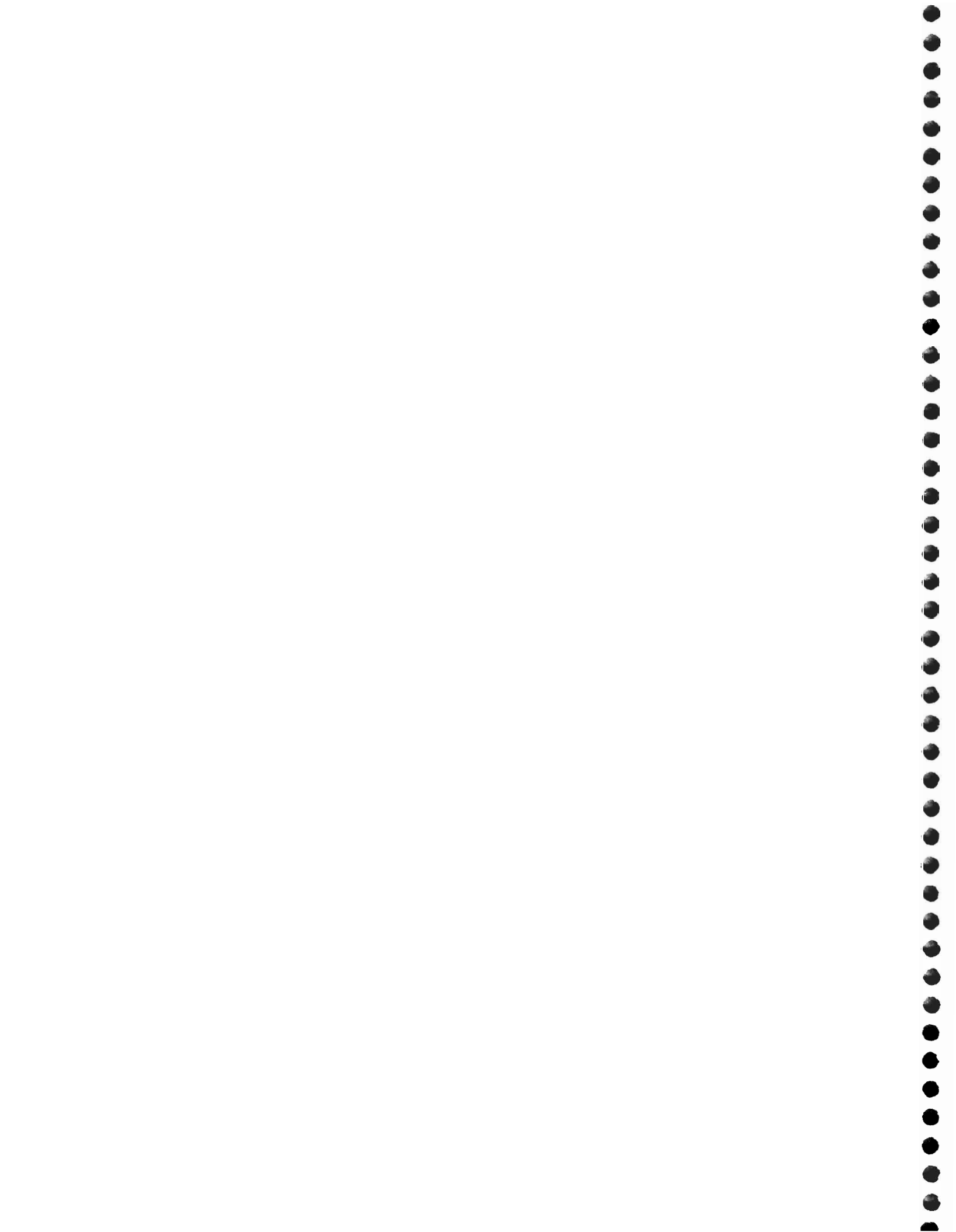
12. Fill in the appropriate commands.

```
>ED INVENTORY _____  
A1  
TOP  
. _____  
'A1' EXITED  
  
A2  
TOP  
. _____  
'A2' EXITED  
  
A3  
TOP  
. _____  
'A3' EXITED  
  
>
```

13. Fill in the commands.

```
>ED INVENTORY *<return>  
A6  
TOP  
. _____  
'A6' EXITED  
  
>
```

NOTES

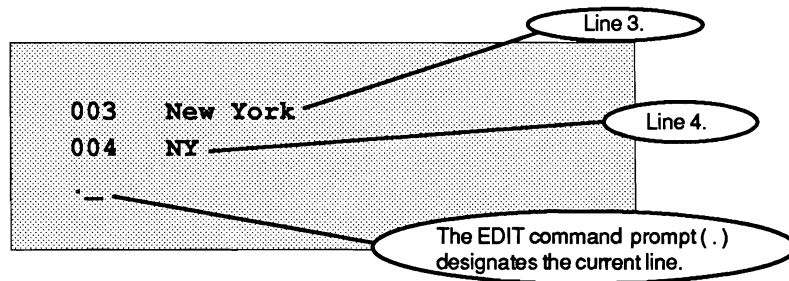


3. Getting Around An Existing Item

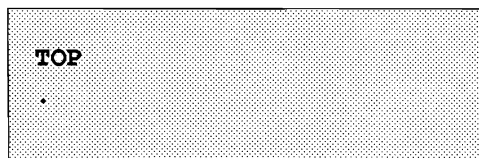
These are the commands which do not affect the data but allow you to move about an item and review the contents. It is necessary to cover these commands before getting into creating and altering items. If you wish to follow along your system, any item in any file will serve to demonstrate the actions of these commands, but don't feel obligated. We're just trying to lay a foundation for Chapter 4, Adding Lines.

EDIT uses an internal line pointer to keep track of the position of the currently active line. All commands which affect the data work on or from the current line position.

For example, if we view a small portion of an item,



The current line is 4.



At the top of the item, the current line is 0.

3.1 Absolute Positioning

Absolute positioning commands take you to a specific point in the item.

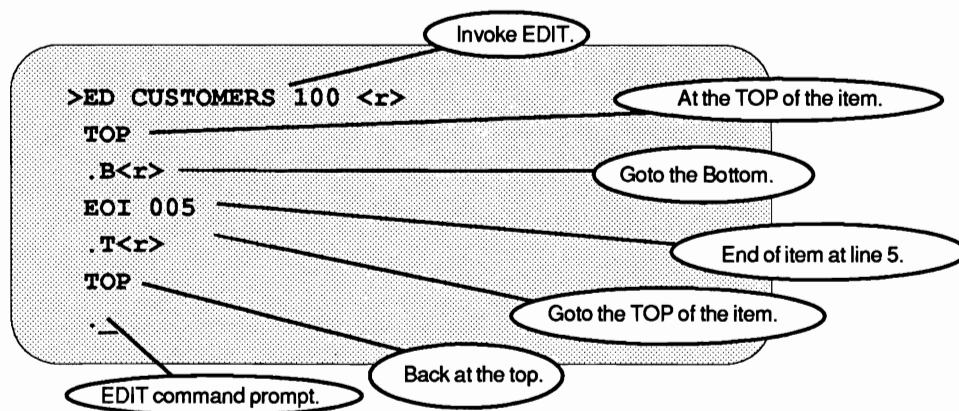
<u>Command</u>	<u>Description</u>
T	Positions the line pointer at the TOP of the item.
B	Positions the line pointer at the BOTTOM of the item.
Line number	Goto line number. The line displays and becomes the current line.
G line number	Goto line number. The G is optional. As above, the line number becomes the current line.

Now, here's Catch 22. You need to be familiar with these commands before you can effectively input data, but we can only demonstrate how they work with existing data. We haven't put the sample data in yet, which comes in the next chapter. So this chapter is FYI. If you're brave, then you can find some existing data on you machine and practice with these commands. These commands will not alter your data.

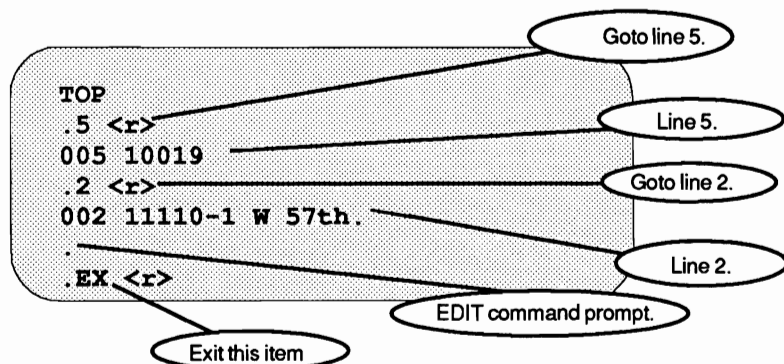
The TOP is just prior to the first line, 001, of the item. It may be considered to have a line number of 000.

The BOTTOM of the item is just after the last line.

Now try this. EDIT item 100 on the CUSTOMERS file.



Either the line number or the G followed by the line number, sets the line pointer to the requested line number.



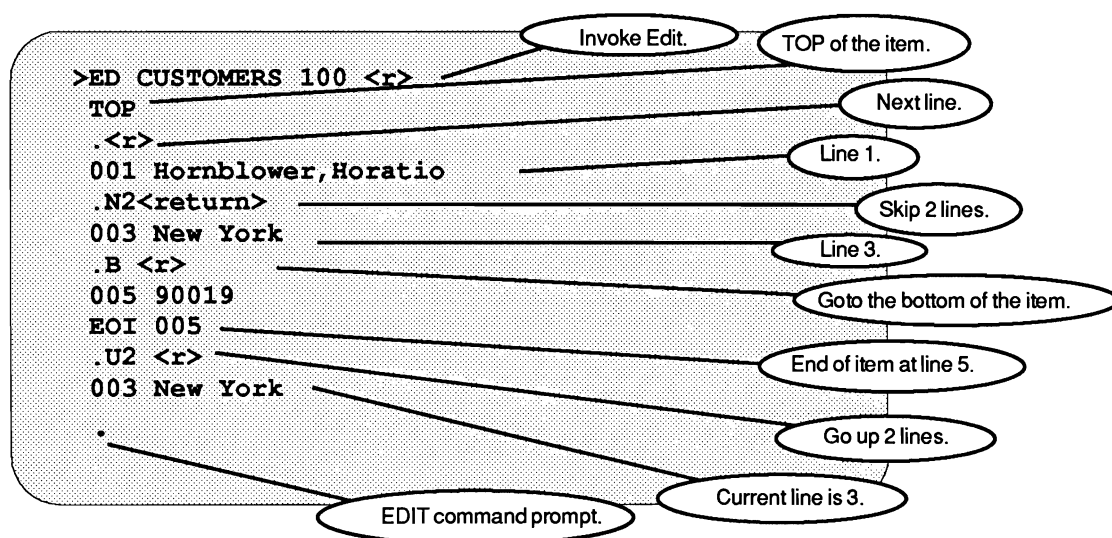
3.2 Relative Positioning

Relative positioning lets you address other lines in the item by skipping forward and backward from the current line.

<u>Command</u>	<u>Description</u>
<return>	Goto and show the next line.
N number of lines	Next number of lines. Go down the next "number of lines" from the current position.
U number of lines	Up number of lines. Go up the "number of lines" from the current position.

(U)p and (N)ext allow you to move the line pointer backward and forward through the item.

EDIT item 100 in the CUSTOMERS file.



3.3 Reviewing The Item

EDIT allows you to move around within the body of an item to display a single line or a series of lines.

The LIST LINES (L) command is used to review portions of the item.

Command	Description
Ln	List (n) number of lines from the current position.

Try this. While still in 100 in the CUSTOMERS file, go to line 2 of item 100 in CUSTOMERS and LIST 3 lines.

```

002 1110-1 W 57th.
.L3 <r>
003 New York
004 NY
005 10019
EOI 005
.

```

The current line is 2.

List 3 lines. (Beyond the current line)

End of item at line 5.

EDIT command prompt.

LIST leaves the line pointer at the position of the last line displayed. Line 5 is now the current line.

Since most terminals have only 24 lines of display at a time, the most you may wish to ask for is 22 lines before the screen will scroll away. The command would be "L22 <r>". Try this command here.

```

TOP
.L22 <re>
001 Hornblower, Horatio
002 1110-1 W 57th
003 New York
004 NY
005 10019
EOI 005
.L3 <r>
003 New York

```

Goto line 3 for the next part of this lesson

3.4 Getting Item Information

The following commands are used for displaying item status information.

<u>Command</u>	<u>Description</u>
?	Show the filename, item-id and present line number.
S?	Show the size of the item. The size of an item is the total number of characters (bytes) which make up the item.

Sometimes it's useful to verify which item you've decided to edit. Try this command here.

```
003 New York
.? <r>
CUSTOMERS 101 LINE 003
.
```

The number of lines in an item is variable and effectively infinite. However, you must remember that the maximum number of characters that an item can hold is 32,000. The "S?" command is useful for checking the growth of an item. Try this command here.

```
.S? <r>
ITEM LENGTH IS 62 BYTES
.
.EX <r> _____
```

There is currently 62 characters
in Item 100.

Now exit to return to TCL.

Exercise 3

1. Item 300 exists on the CUSTOMERS file. Fill in the TCL command to invoke EDIT. Also fill in the proper system response.

```
> _____  
_____  
.
```

2. EDIT keeps track of the current line, using a _____ .

3. The display of a line number and the value of the line followed by a period,

```
003 Los Angeles
```

```
.
```

designates that the current line is _____ .

4. Using absolute positioning, fill in the appropriate commands.

```
. _____  
EOI 005  
.  
002 12 Anyway Street  
.  
004 CA  
.  
TOP  
.  
'300' EXITED  
>
```

5. Using relative positioning, fill in the appropriate commands.

```
. _____  
EOI 005  
.  
001 Kvetch, Philo  
.  
004 CA  
.  
005 97711  
,EOI 97711  
.
```

The current line is _____ .

Exercise 3 - page 2

6. Still in the same item, fill in the appropriate commands.

. _____
TOP

. _____
001 Kvetch, Philo
002 12 Anyway Street
003 Yentaville
004 CA
.

The current line is _____ .

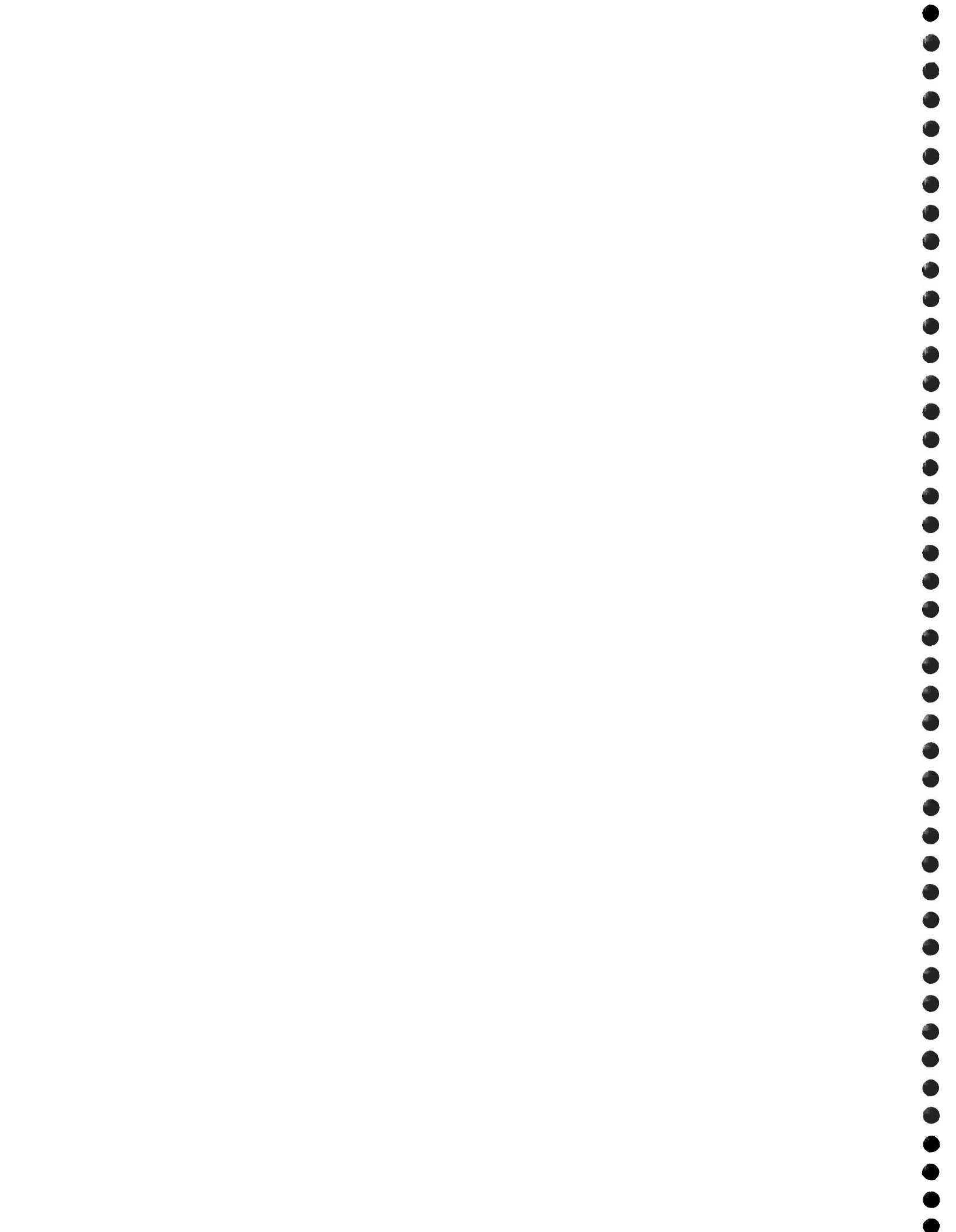
7. The command to check the item size is _____ .

8. The maximum number of lines in an item is _____ .

9. In this same item, fill in the system response.

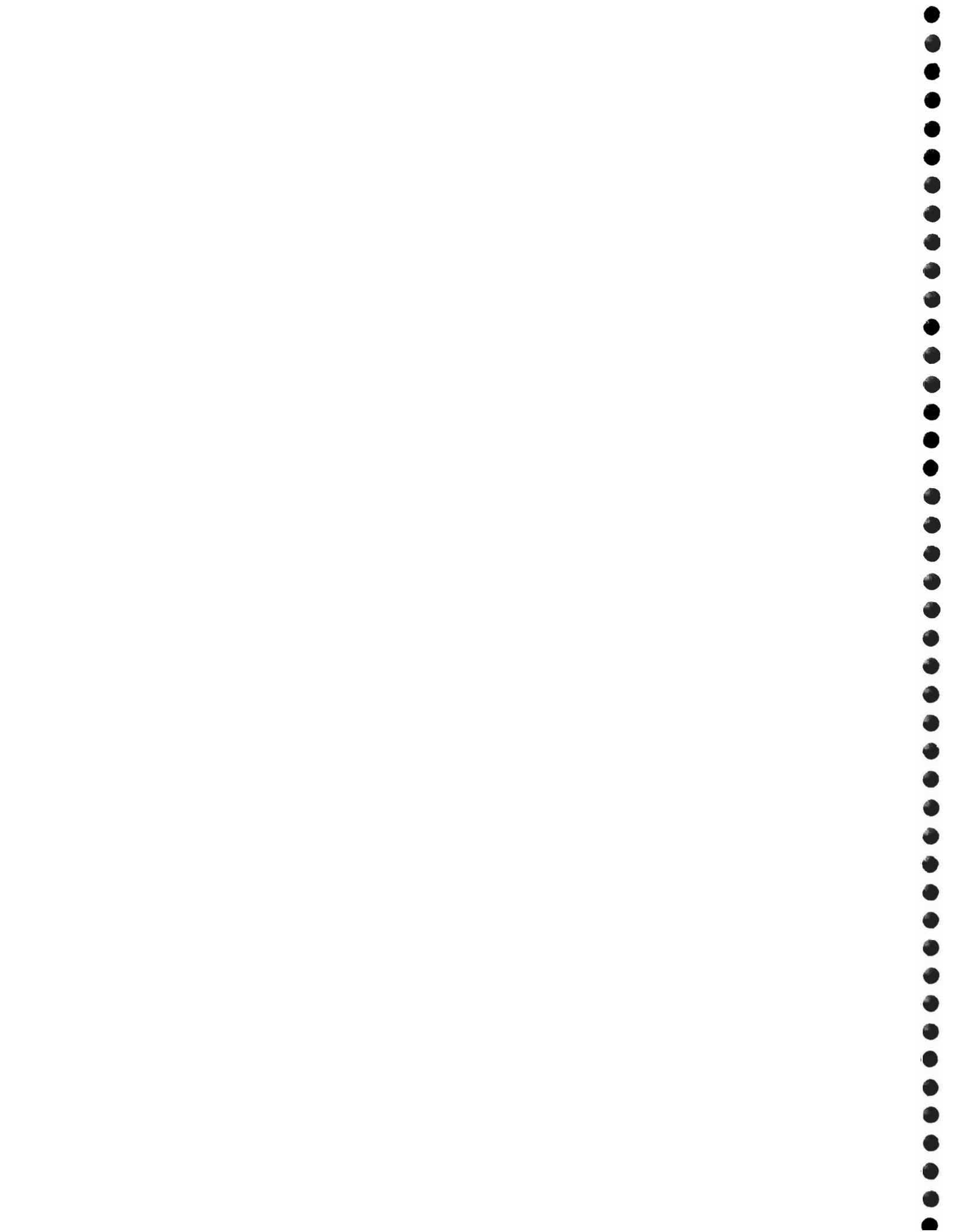
004 Yentaville
.? <return>

10. What command reviews a screen's worth of an item? _____





NOTES

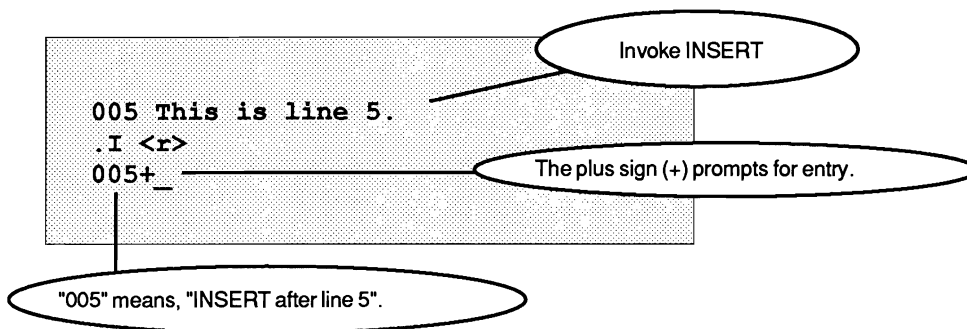


4. Adding Lines

The process of adding lines to a new or an existing item is called INSERT.

<u>Command</u>	<u>Description</u>
I	Continuous INSERT. Allows the entry of multiple lines of data. A <return> by itself on a line ends INSERT.
I text	INSERTS the single line of text following the current line.
Related Command	
F	Flip buffers. Include the last series of changes to the work item without writing to the file. This occurs automatically at the termination of INSERT in a new item. See "Saving Your Work."

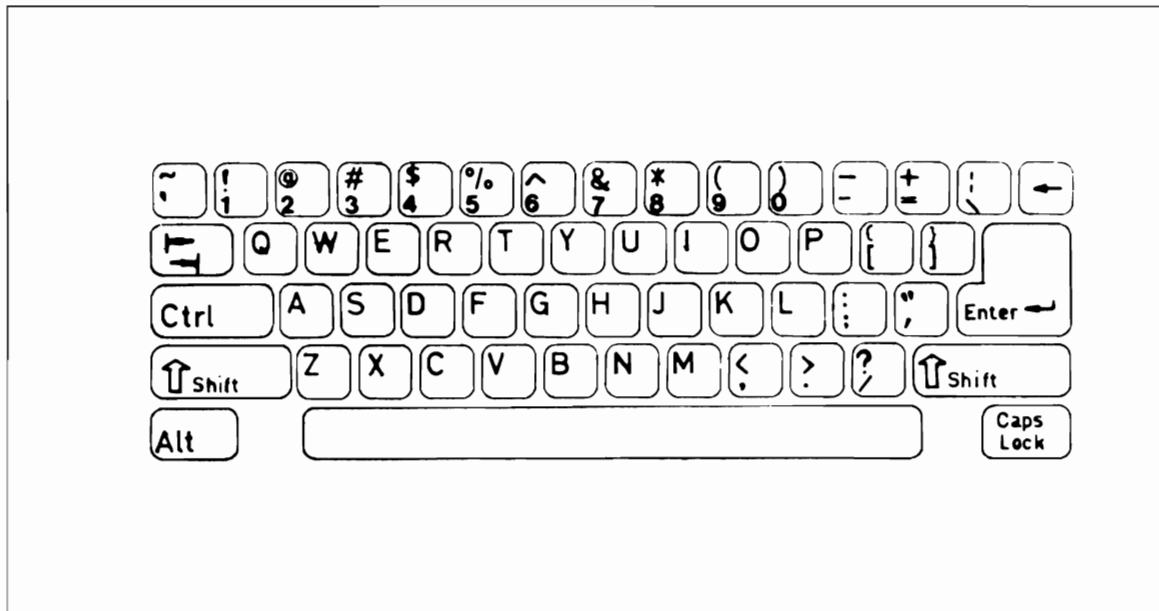
Invoking the I command prompts you to enter data from the terminal's keyboard. The INSERT prompt is the plus sign (+). It would appear on your terminal as in this example:



EDIT is now waiting for you to enter text from the keyboard.

4.1 Entering Text From The Keyboard

You can now enter text and data from the terminal keyboard. While INSERT is active, all keystrokes are treated as text, except for the following "control" keys:



Special Keys

- <return>** indicates the end of a line.
- <backspace>** back up input by 1 character. You can't <backspace> beyond the beginning of a line.
- <tab>** advances input to the next tab stop.
- <shift>** is the Shift key. Used to change the case of letters or the value of number and punctuation.

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: 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 list covers what is displayed on most terminals. However, it is not standard and may differ on yours. If this is true in your case, note the characters displayed for future reference. Now we're ready to create some test data using the editor.

<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 multi-value delimiter, a Value Mark. 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 (\).

Entering Attribute Marks causes a new EDIT line to be generated since attributes indicate "end of line." (See "ADDING NULL LINES.")

The Value and Sub-Value Marks are held within an attribute and display on the EDIT line as described.

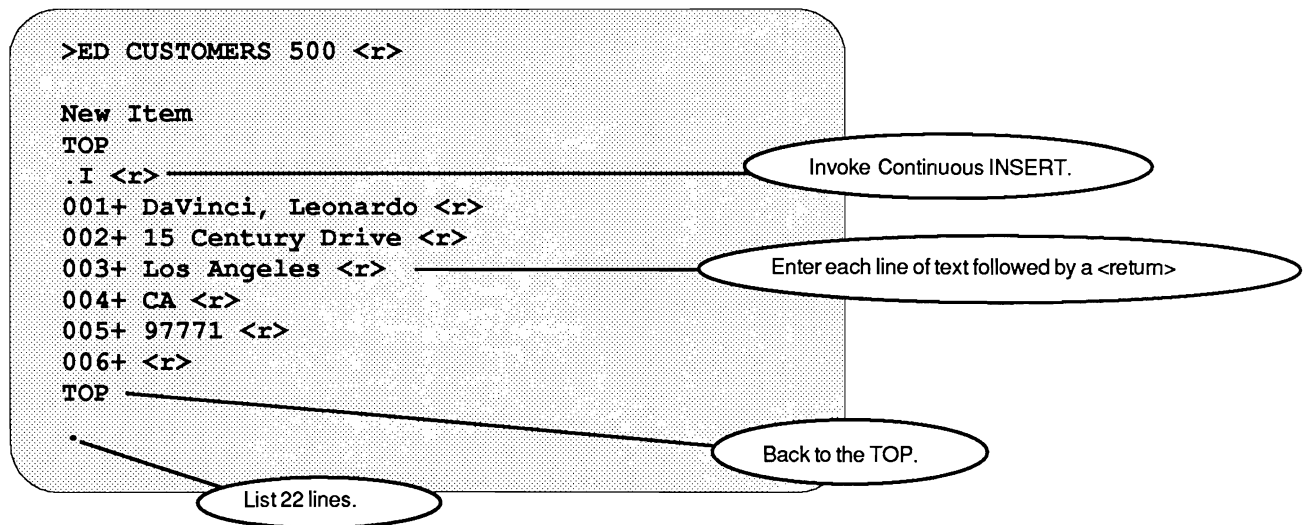
4.2 Adding New Items To A File

This example creates an item in the CUSTOMERS file using the following item layout:

```
Item-Id is a number: 100, 500 etc.  
Attribute 1 : Last Name, First Name  
Attribute 2 : Address  
Attribute 3 : City  
Attribute 4 : State Abbreviation (2 characters)  
Attribute 5 : Zip
```

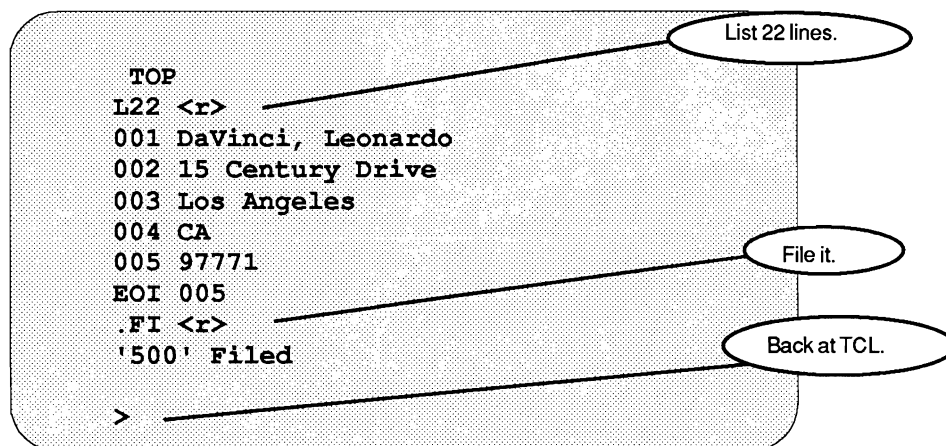
Add customer 500 to the CUSTOMERS file. Remember, INSERT adds lines immediately following the current line position.

Invoke EDIT:



This is an example of a continuous INSERT. INSERT continues until the return key is pressed at the first position of a line. The return key pressed at line 006 does an automatic Flip and resets the pointer to the TOP of the item. This occurs only in NEW items. Notice that the lines are being automatically numbered sequentially.

Review this item:



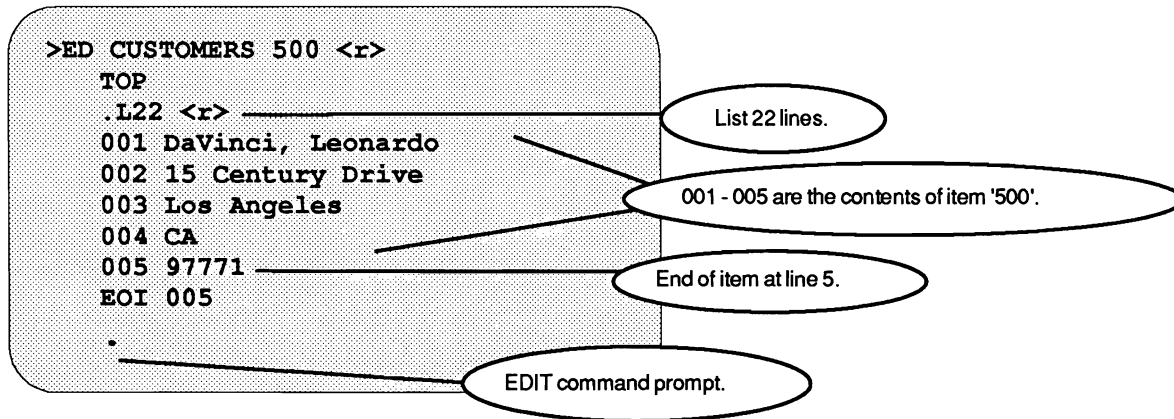
The FILE (FI) command saves the item to the file and exits the Editor. (See "SAVING YOUR WORK.")

If the EXIT (EX) command had been typed, EDIT would have exited the item before it could have been saved.

4.3 Adding Lines to an Existing Item

Now, you need to know how to insert new lines in an existing item. Again, INSERT adds lines immediately following the current line position.

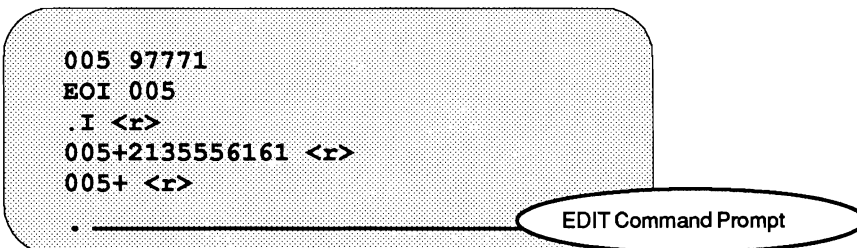
At the risk of sounding redundant, invoke EDIT again for item 500 and review it. (Ask to list 22 lines.)



There are only 5 lines in this example. Even though the list requested is 22 lines, listing stops at the bottom of the item. "EOI" stands for "end-of-item."

Add a phone number on line 006. In other words, INSERT a line after 005. Since the pointer is already on line 005, there is no need to goto the line.

Using continuous INSERT:



Sequential numbering during INSERT occurs only when a new item is being created. In all other cases, the current line number is repeated for each prompt. The "005+" prompt means "to be INSERTed after line 5".

In this example, only one line is added. An alternative in this case is to use the single line INSERT:

005 97771
EOI 005
.I 2135556161 <r>
.

EDIT Command Prompt.

This method is a little quicker way of adding 1 line.

The phone number is now line on 6.

After entering any series of changes to an item, the command (F) must be invoked. (F) or "Flip" includes the last changes and rennumbers the added lines in the proper sequence. (See "SAVING YOUR WORK.")

.F <r>
TOP
.

Flip the buffers.

Review the item.

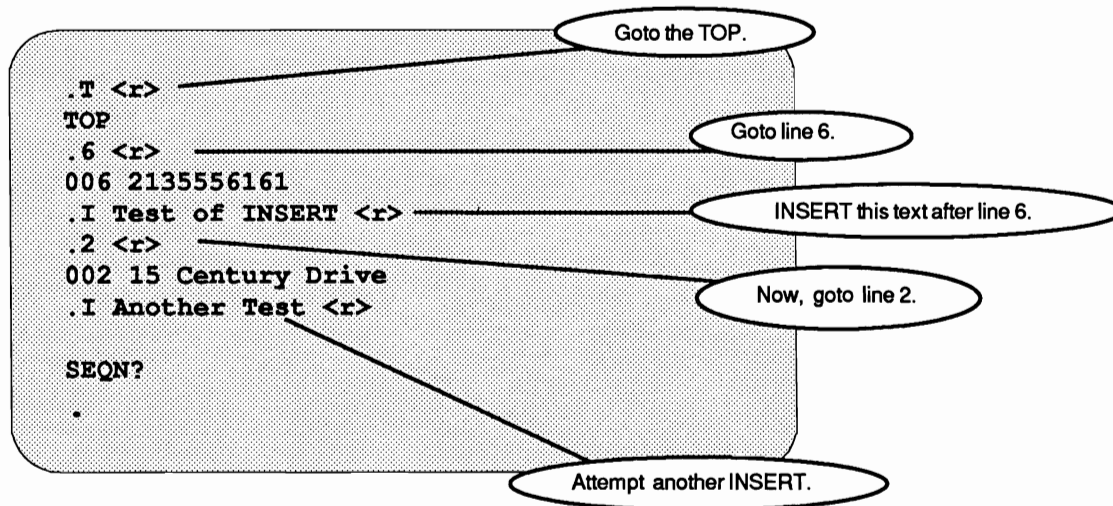
.L22 <r>
001 DaVinci, Leonardo
002 15 Century Drive
003 Los Angeles
004 CA
005 97771
006 2135556161
EOI 006
.

List 22 lines.

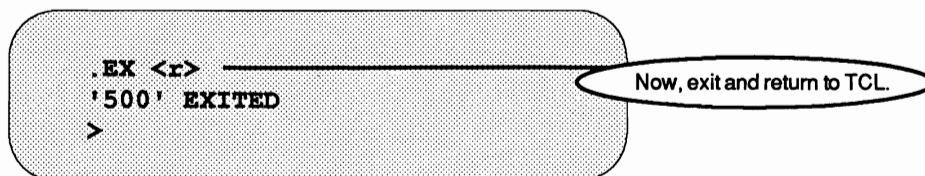
End of item at line 6.

EDIT Command Prompt.

The sequence of INSERTs, as in all changes, must be from the TOP of the item down. Changes must be made in ascending sequence of line numbers. If you enter out of sequence, the "SEQN?" error message appears. (F)ipping the buffer lets you start from the top down again.



The second INSERT was performed at line 002 after inserting at line 006. The (F)lip command was not invoked. No way Jose!



4.4 Adding Null Lines

If you are using EDIT for documentation, or creating data definition items, you may need to skip lines. This is referred to as imbedding a null line.

A null line is an empty line. However, INSERT inherently does not allow null lines to be entered because entering <return> at the FIRST position of a line terminates the INSERT session.

There are two methods for imbedding null line while in INSERT.

4.4.1 Null Lines, Method 1

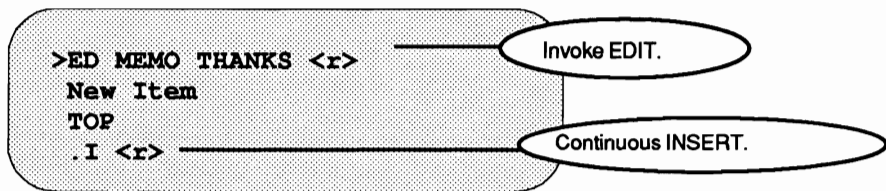
Choose any single character as a line holder. Make sure that this is a unique character, something you don't find in the rest of the item.

The REPLACE LINES (R) command is used to replace these characters with a null, hence creating an empty line.

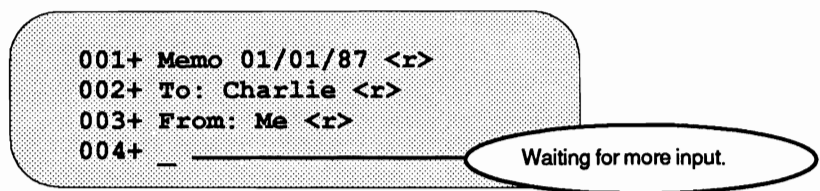
The Commands involved are:

<u>Command</u>	<u>Description</u>
I	Continuous INSERT. See the section entitled "ADDING LINES".
R/oldstring/newstring/	REPLACE the first occurrence of oldstring on a line with newstring. (See "REPLACING", Page ##.)
F	Flip buffers. Reflect the last series of changes to the work item without writing to the file. This occurs automatically at the termination of INSERT in a new item. (See "SAVING YOUR WORK", Page ##.)

Create the new item THANKS in the MEMO file.



Enter the following text line by line from the terminal keyboard.



A NULL line is needed. If the <return> key is pressed, INSERT will end. For this example, use the character "X" as a line holder.

```
004+ X <r>
005+ Charlie, <r>
006+ This is to inform you that your gift was received <r>
007+ and greatly appreciated. A very merry to you too! <r>
008+ X <r>
009+ Thanks, <r>
010+ <r>
TOP
.
```

"X" is the holder.

Another line holder at line 8.

End continuous INSERT.

EDIT command prompt.

TOP of the item, Automatic (f)lip.

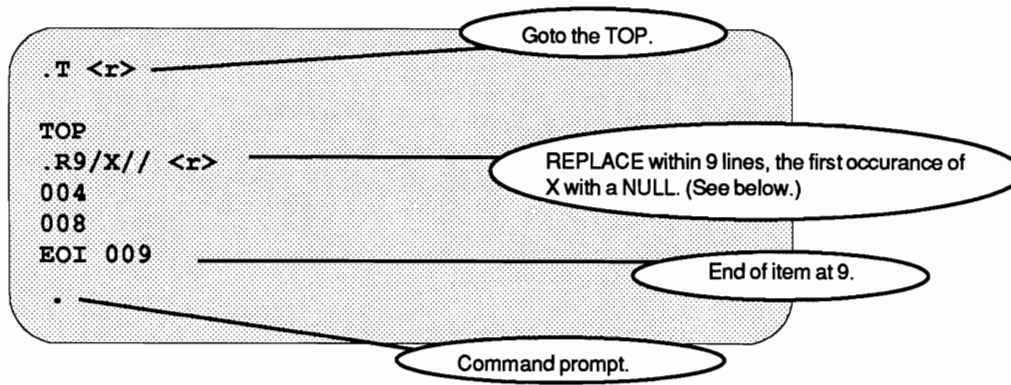
Review the item.

```
.L22 <r>
001 Memo 01/01/87
002 To: Charlie
003 From: Me
004 X
005 Charlie,
006 This is to inform you that your gift was received
007 and greatly appreciated. A very merry to you too!
008 X
009 Thanks,
EOI 009
.
```

List 22 lines.

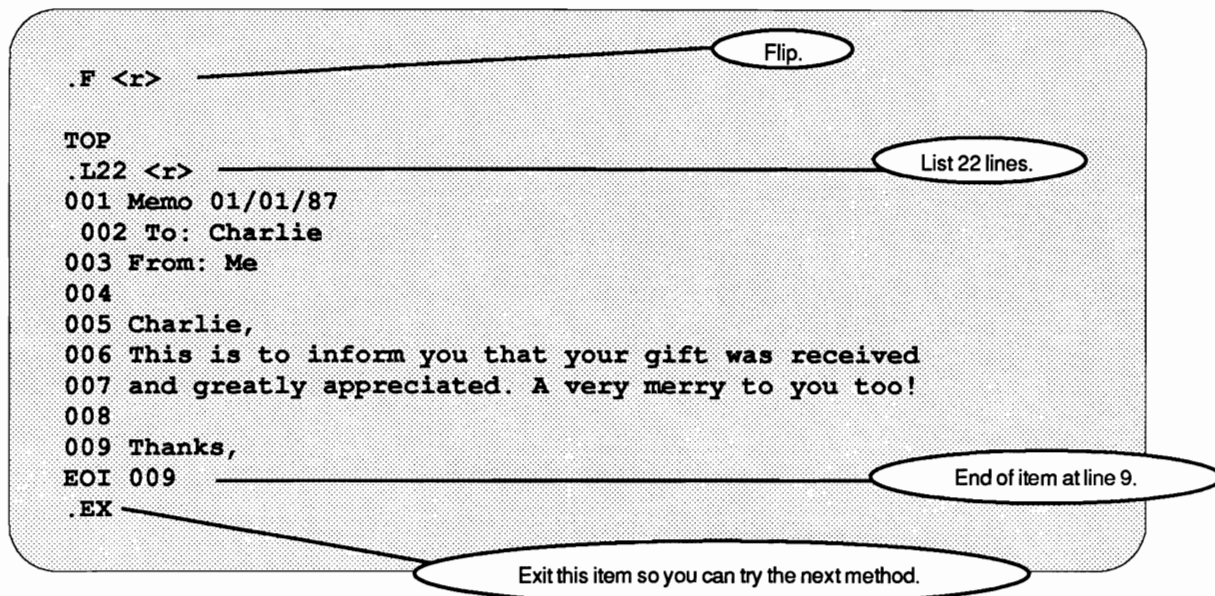
End of item at line 9.

Now, use "R" (REPLACE) to remove the X's and place NULLS on the line as follows;



Both lines 4 and 8 contained "X." Since "X" was the only character on these lines, the lines were set to NULL. Flip the buffer and review the changes. You have to do this because the initial INSERT session is ended. This item is no longer considered new.

Note: In the REPLACE command, two consecutive delimiters (//) indicate NULL. (See "REPLACING.")



4.4.2 Null Lines, Method 2

The next method of inserting null lines is performed by imbedding actual attribute marks within the newly inserted text.

Remember, attributes are the characters that the system uses to indicate the end of a line.

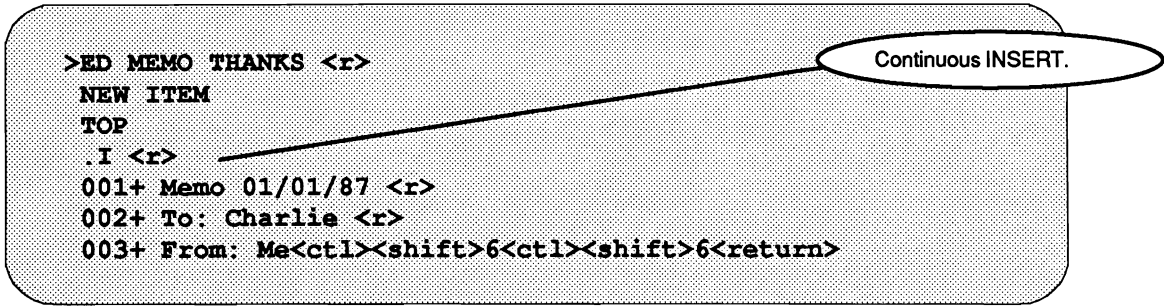
Attribute marks are generated from the keyboard by pressing the <ctl> key simultaneously with the up-arrow (^) character. On most keyboards, the "^" is a "<shift>6."

To generate an attribute mark from the keyboard, press:

<ctl><shift>6. (simultaneously)

A NULL line can be generated as follows,

Start over using Method 2.



```
>ED MEMO THANKS <r>
NEW ITEM
TOP
.I <r>
001+ Memo 01/01/87 <r>
002+ To: Charlie <r>
003+ From: Me<ctl><shift>6<ctl><shift>6<return>
```

Continuous INSERT.

The two attribute marks after the word "me" are imbedded to delimit a NULL line. Attributes appear on the display as up-arrows (^).

The line displays on the terminal as follows:

```
003+ From: Me^^ <r>
```

At this moment, line numbering is out of sequence, since line 3 as displayed actually contains 2 attributes.

Continue entering.

```

004+ Charlie, <r>
005+ This is to inform you that your gift was received <r>
007+ and greatly appreciated. A very merry to you!^^ <r>
008+ Thanks, <r>
009+ <r>
TOP
.

```

Another NULL line is added in the same way with two attribute marks.

A review of the item shows that all attributes are now at their correct line positions

```

.L22 <r>
001 Memo 01/01/87
002 To: Charlie
003 From: Me
004
005 Charlie,
006 This is to inform you that your gift was received
007 and greatly appreciated. A very merry to you too!
008
009 Thanks,
EOI 009
.FI

```

List 22 lines.

End of item at line 9.

File this item.

4.5 Setting Tab Stops

It is often necessary to align input at predetermined margins. This is accomplished by setting tab stops. Tab Stops are set by the (TB) command.

<u>Command</u>	<u>Description</u>
TB n n n ..	Set tab stops. Each n represents the columnar position where the tab stop is placed.
Related Command;	
C	Display columnar positions. Useful for determining tab stop columnar positions.

During INSERT the <tab> key (<ctl>I on keyboards without a <tab>) positions the cursor for entry at the next tab position. Pressing the <tab> key imbeds spaces to the next stop rather than control characters as do word processors.

Setting tab stops is quite useful when indenting BASIC source statements.

Note: Tab stops are remembered only within the same EDIT session, whether Editing a single item or a series of items from an Item-List.

Each time EDIT is invoked, tab stops must be redefined.

Try this example:

The diagram illustrates the sequence of commands for editing a file and setting tab stops. Callouts provide context for each step:

- Edit Item 'TEST' in the file 'SAMPLES'.** points to the command: `>ED SAMPLES TEST <r>`
- Display Columnar positions.** points to the command: `.C <r>`
- Set tab marks at column positions. 10, 20, 30 and 40** points to the command: `.TB 10 20 30 40 <r>`
- File this.** points to the command: `.FI`

The output of the `.C` command shows column positions 1 through 40. The output of the `.TB` command shows the tab marks set at 10, 20, 30, and 40. The output of the `.I` command shows the test text with tabs at the specified positions.

```
>ED SAMPLES TEST <r>
New Item
Top
.C <r>

      1      2      3      4
123456789012345678901234567890123456789012345678
.TB 10 20 30 40 <r>
.I <r>
001 *This is a test of Tab positions <r>
002 <tab>    at the first tab stop <r>
003 <tab>    <tab>    at the second <r>
004 <tab>    <tab>    <tab>    the third <r>
005 <r>
Top
.FI
```

If you wish, use EDIT as described above and add the following "Sample Items" to their respective files. These sample items will be used in the remaining chapters.

Sample Items

Items to be added to the file CUSTOMERS:

ID:100

001 Hornblower, Horatio
002 1110-1 W 57th
003 New York
004 NY
005 10019

ID:200

001 Bunny, Bugs
002 0-0 Hole-in-the-ground
003 Brooklyn
004 NY
005 10101

ID: 300

001 Participle, Dangle
002 3 Dat Street
003 Grammar City
004 PA
005 12345

ID: 400

001 Gardner, Chauncy
002 12 Main St
003 Washington
004 DC
005 00212

ID: 500

001 Da Vinci, Leonardo
002 15 Century Drive
003 Los Angeles
004 CA
005 97771

ID: 700

001 Phink, Clyde
002 220 Culver
003 Irvine
004 CA
005 97215

ID: 1000

001 Fudd, Elmer
002 30 Wabbit Way
003 Wedondo Beach
004 CA
005 97000

Add to the file MEMO:

ID: THANKS

001 Memo 01/01/87

002 To: Charlie

003 From: Me

004

005 Charlie,

006 This is to inform you that your gift was received

007 and greatly appreciated. A very merry to you too!

008

009 Thanks,

Add to the file SAMPLES:

ID: ATEST

001 This is the first line.

002 This is the second line.

003 This is the third line.

004 This is the fourth line.

005 This is the fifth line.

006 This is the sixth line.

ID: ANOTHERTEST

001 This is an example of a wildcard

002 search. Wildcards are designated

003 by the up-arrow (^). This should

004 wind up the discussion of

005 wildcards.

Build data in the sample file.

At TCL enter:

> Copy samples ATEST ATEST ATEST ATEST ATEST ATEST ATEST ATEST <r>

TO: A B C D E F G <r>

7 items copied

>

Exercise 4

1. Match the keys and their functions.

_____ [	a. Generates an Attribute Mark
_____ <return>	b. Back up 1 character
_____ <ctl>Q	c. <escape> displays as this
_____ <ctl><shift>6	d. X-ON
_____ ^	e. Same as <backspace>
_____ <backspace>	f. Advance to next tab stop
_____ <ctl>W	g. Re-enter entire line
_____ <ctl>X	h. X-OFF
_____ \	i. Terminates an INSERT session
_____]	j. Attribute Mark displays as this
_____ <ctl>S	k. Value Mark displays as this
_____ <ctl>I	l. Back up input by 1 word
_____ <ctl>H	m. Sub-Value Mark displays as this

2. Fill in the commands to create the following item in the SAMPLES file:

ID: EXERCISE4

001 This is a test of
002 how much you have
003 learned in the ADD LINES
004 Section of The Editor
005 Workbook.

The end-of-item is at 005.

```
> _____  
NEW ITEM  
TOP  
.  
001+ _____  
002+ _____  
003+ _____  
004+ _____  
005+ _____  
006+ _____  
  
TOP  
.
```

Exercise 4 - page 2

3. What command would you use to review the item? _____
4. The item ATEST already exists in the SAMPLES file. Fill in the blanks to add 2 lines after line 2. Use any text that you wish.

```
> _____
TOP
.
001 This is line 1.
002 This is line 2.
003 This is line 3.
004 This is line 4.
.
002 This is line 2.
.
002+ _____
002+ _____
002+ _____
.
TOP
.
```

5. Fill in the command which saves the item.

```
. _____
'TEST' FILED
>
```

6. Explain what went wrong.

```
>ED SAMPLES ANOTHERTEST <return>
Top
.5 <return>
005 This is test line 5
.I INSERT this <return>
.G1 <return>
001 This is the first line
.I INSRET this also <return>
SEQN?
.
```

Exercise 5

1. A Null line is _____.
2. INSERT is terminated by entering a _____ at position _____ of a line.
3. What is a line holder? _____.
4. Add 2 NULL lines after line 4 of the item WELCOME in the MEMO file.
Using Method 1, fill in the appropriate commands.

```
> _____  
TOP  
.  
004 Call me if you need me.  
.  
004+ _____  
004+ _____  
004+ _____  
.  
_____
```

TOP

.

Fill in the commands to remove the line holders.

```
. _____  
005  
006  
EOI 010  
.
```

5. Use Method 2 to achieve the same thing.

```
> _____  
TOP  
.  
004 This is line 4  
.  
_____  
.  
_____  
  
TOP  
.
```

Exercise 6

1. What is the command to set tab stops? _____
2. What keys advance input to the next tab stop? _____ or _____
3. The command to display columnar positions is _____ .
4. Item TABTEST is in the SAMPLES file.

ID: TABTEST

001 Name	AMOUNT	DATE
002	DUE	DUE
003 John	100	01-01-87
004 Mary	150	02-15-87
005 Sasha	300	03-01-87

Add line 6 for Jason who owes 250 by 06-01-87.

> _____
TOP

. _____
005 Sasha 300 03-01-87
. _____

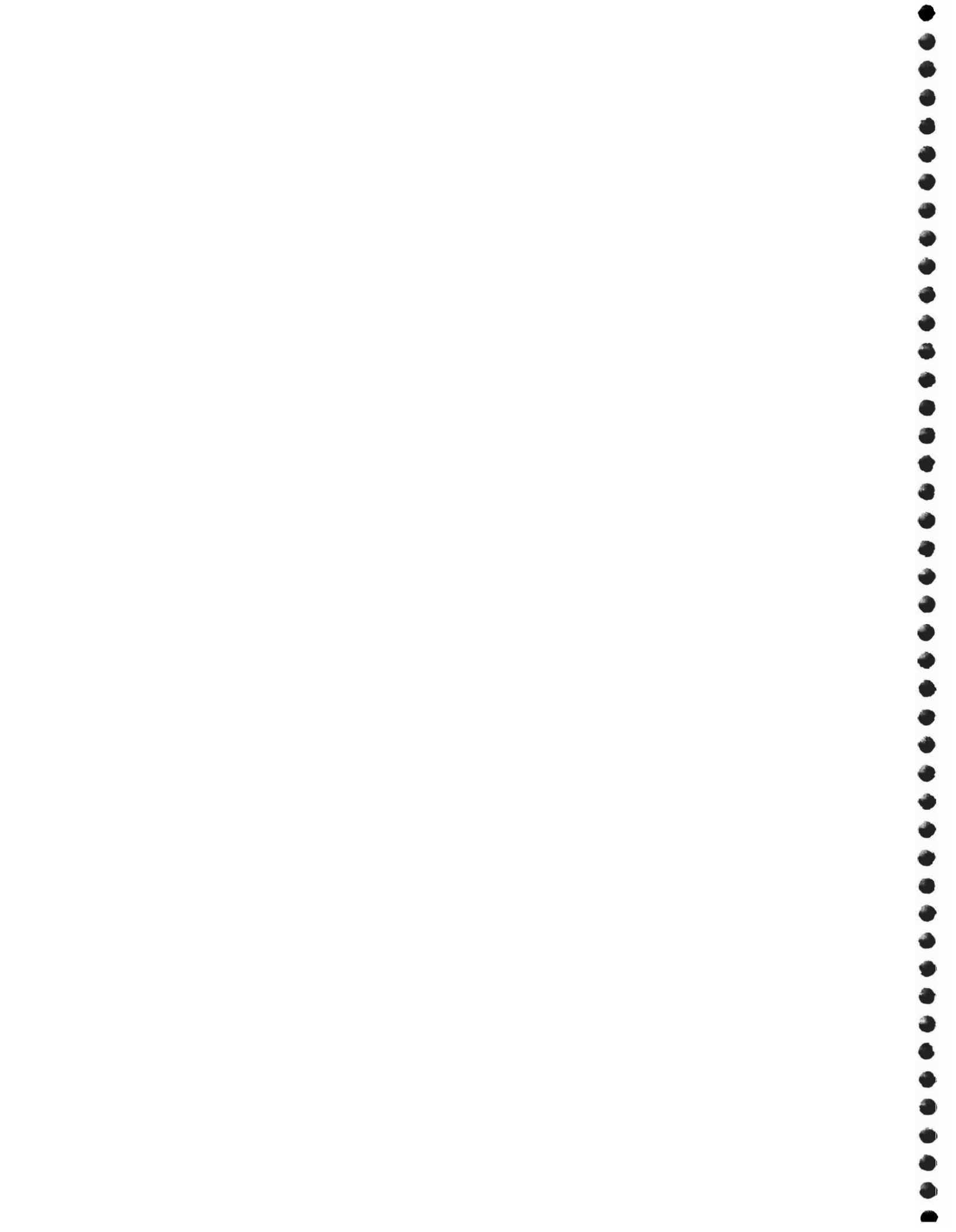
1234567890123456789012345678901234567890....

Now set the tab stops and INSERT the line.

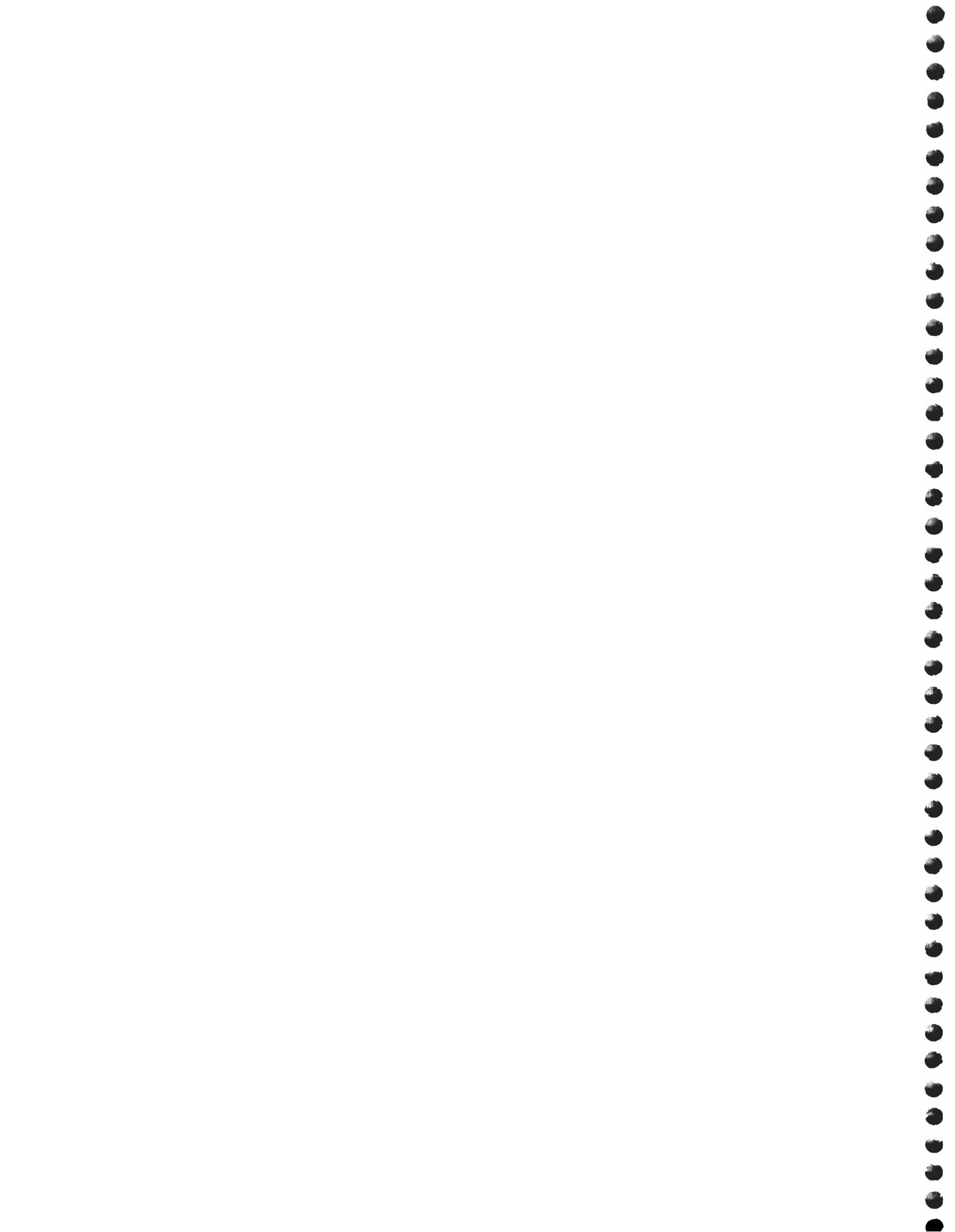
. _____
. _____
005+ _____
005+ _____
. _____

TOP

.



NOTES



5. Saving Your Work

The FILE SAVE commands cause the EDIT work item to be written to a file on disk. The process is called FILING the item. The work item can be overwrite to the original item or be written to a new item in the same or a different file.

5.1 The Work Item

When EDIT retrieves an item, it works with it in a temporary work area called a buffer or "scratchpad." EDIT actually uses two work buffers. Each takes turns containing the current and changed versions of the item. Think of these as "Work Items."

<u>Command</u>	<u>Description</u>
F	Flip buffers. Reflect the last series of changes to the work item without writing to the file. This occurs automatically at the termination of INSERT in a new item.
Related Commands;	
XF	Cancel all changes made since the last (F)lip. (See "CANCELLING CHANGES.")
X	Cancel the last change.

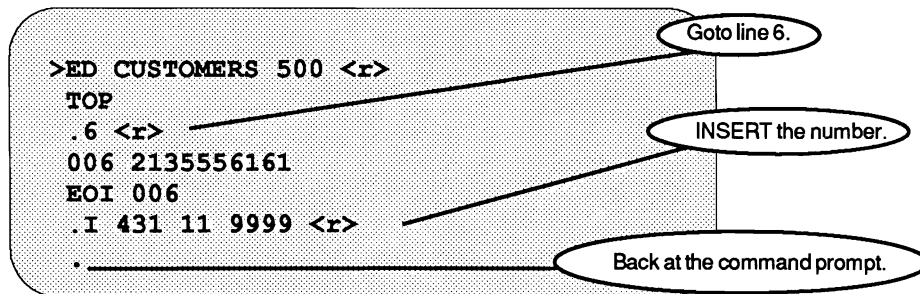
Invoking the F command tells EDIT to include all changes in the work item made since the last (F) was invoked. This is called "flipping" the buffers.

The original item remains unaltered until an (FI) or (FS) command is performed. (See next example.)

Take item 500 on the file CUSTOMERS.

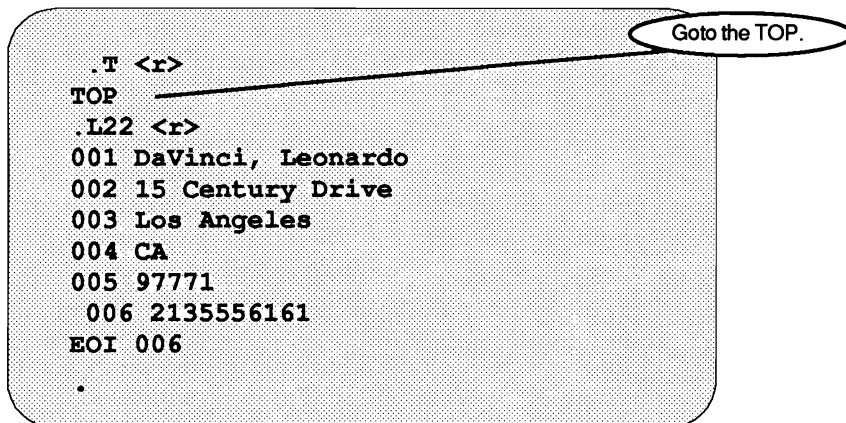
Put a Social Security Number on line 7 of the item.

Invoke EDIT:



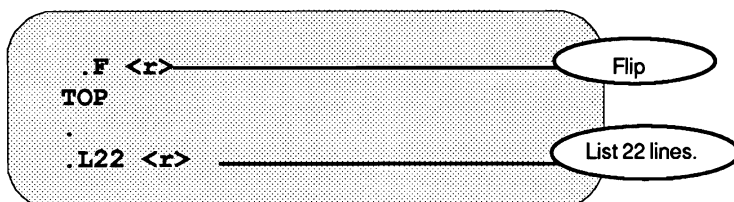
The number 431 11 9999 is now in attribute 7 and should be displayed on line 007.

However, on review, the change is not reflected.

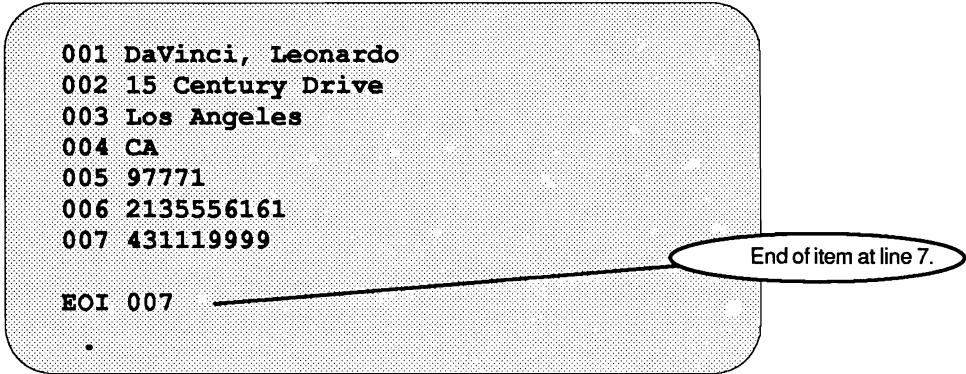


The new line is apparently missing. Actually, the change hasn't been included in the active work item.

Flip the buffer by entering:



Review the change on the previous item.

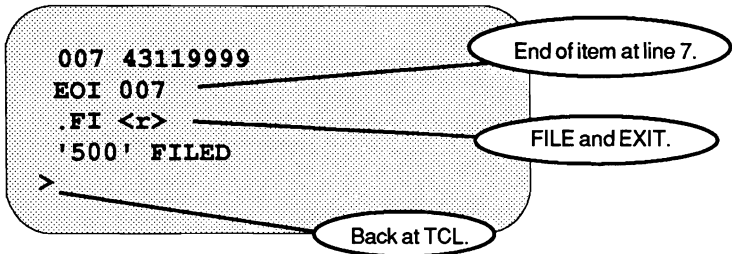


5.2 Saving an Item to the Same Item-id

<u>Command</u>	<u>Description</u>
FI	FILE the item and exit EDIT. If an item-list is active, EDIT the next item.
FIK	FILE the item, exit EDIT and KILL the list.
FS	FILE the item, stay in EDIT and go to the TOP of the same item.

Invoke FI to FILE and overwrite the item and EXIT the EDIT process.

Changes have been made to this item. We want to FILE and EXIT.



The item is filed and you are returned to TCL.

If an item-list is active, the next item is edited when the (FI) is invoked.

EDIT 4 items in the CUSTOMERS file.

```
>ED CUSTOMERS 500 700 1000 200 <r>
```

```
500
```

```
TOP
```

```
.
```

500 is the first item.

At the TOP.

For this example, pretend changes have been made and file the item.

```
TOP
```

```
.FI<return>
```

```
'500' FILED
```

```
700
```

```
TOP
```

```
.
```

FILE and EXIT

700 is the next.

At the TOP.

EDIT command prompt

Item '700' is automatically edited. In this case, the rest of the items on the list are to be skipped. FIK saves the current item and KILLS the active item-list, returning control to TCL.

```
700
```

```
TOP
```

```
.FIK <r>
```

```
>_
```

FILE, KILL the list, and exit to TCL.

Back at TCL.

Often, you may want to save your work without terminating the edit session. The FS command allows you to do this. Now try this:

```
ED CUSTOMERS 500
```

```
005 97771
```

```
EOI 005
```

```
.FS <r>
```

```
TOP
```

```
.
```

End of item at line 5.

FILE the item.

TOP of the same item.

EDIT command prompt.

The item has been written back to the file and we are at the TOP of the item, ready for more changes. It's usually a good idea to perform an FS periodically while you work. This insures that all your changes made up to this point are saved. If an EX is invoked unintentionally before you have FILEd the item, all changes would be lost.

5.3 SAVING AN ITEM TO A DIFFERENT ITEM-ID

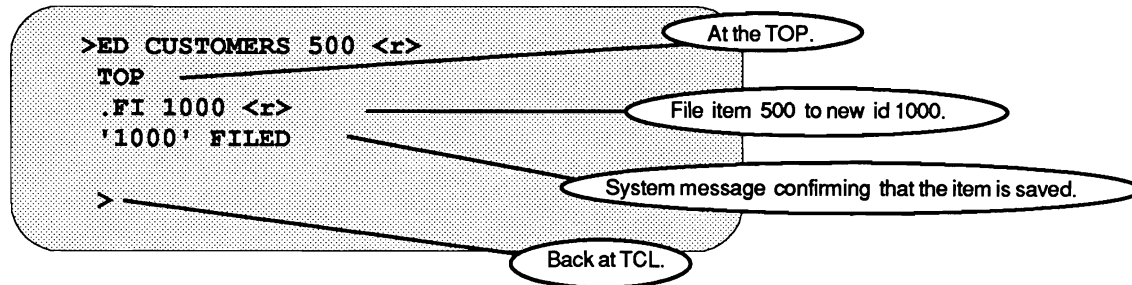
Invoking the FILE SAVE command overwrites by default the original item on file with the edited work item. There are advantages to saving your changes to another item in the same or a different file while leaving the original copy intact. This eliminates the "oops, I didn't want to do that" situation.

The following commands file the item under a new item-id within the same file or in a different file.

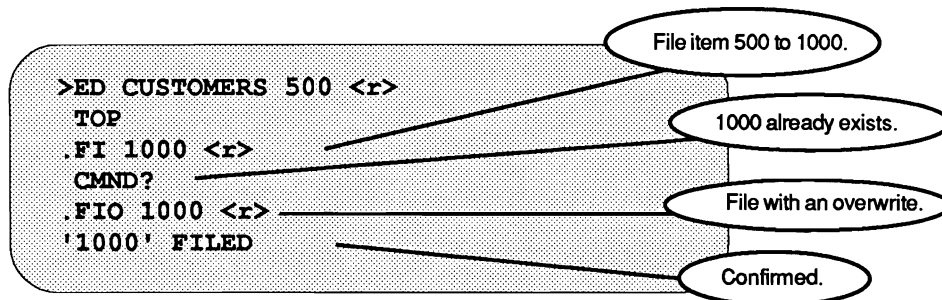
<u>Command</u>	<u>Description</u>
FI(K) item-id	FILE the work item under a new item-id on the same file and exit EDIT. This will not work if the new id already exists on file. The (optional) "K" means KILL any active item-list and return to TCL.
FI(K)(filename item-id	FILE the work item to a new item in a different file and exit EDIT. This will not work if the new item-id already exists. If an item-list is active, EDIT the next item. The (optional) "K" means KILL any active item-list and return to TCL.
FI(K)O item-id	FILE the work item under a new item-id overwriting the new item if it already exists and exit EDIT. If an item-list is active, EDIT the next item. The (optional) "K" means KILL any active item-list and return to TCL.
FI(K)O(filename item-id	FILE the work item under a new item-id in the specified file overwriting if the same item-id already exists in that file, and EXIT. If an item-list is active, EDIT the next item. The (optional) "K" means KILL any active item-list and return to TCL.
FS item-id	FILE the work item to a new item-id on the same file and stay in EDIT. This will not work if the new id already exists on file.
FS(filename item-id	FILE the work item under a new item-id in the specified file and stay in EDIT. This will not work if the new id already exists.
FSO item-id	FILE the work item under the new item-id overwriting the new item if it already exists, and stay in EDIT.
FSO(filename item-id	FILE the work item under a new item-id overwriting the new item on the specified file and stay in EDIT.

When saving to a new file, the item-id specification is optional. If not specified, the original item-id is used by the system. After an FS command, the line pointer is reset to the TOP of the item.

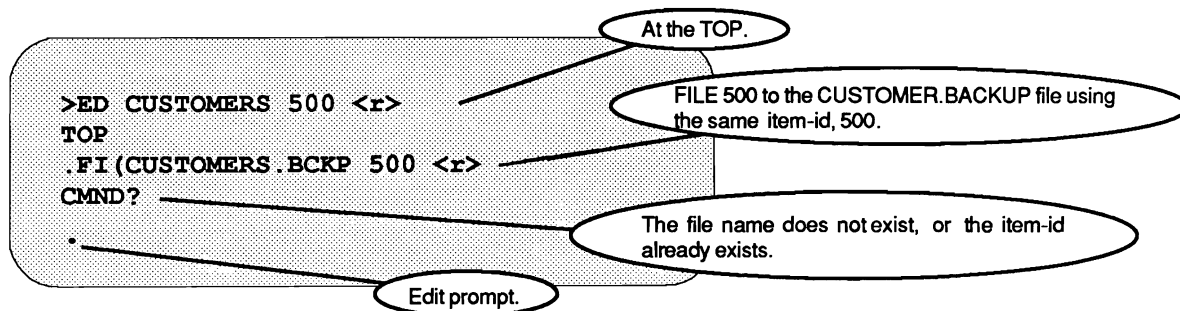
Invoke EDIT for item 500 on the CUSTOMERS file.



Invoke EDIT again:



When the new item already exists, you must use the "0" option to force an overwrite. "CMND?" means the command cannot be processed. This can also happen if you try to save to a file that does not exist.



In this case, the file name was misspelled.

Try again.

```
>ED CUSTOMERS 500 <r>
TOP
.FI (CUSTOMERS.BACKUP 500 <r>
'500' FILED
>
```

(FS) works the same way, except that EDIT remains active on the item.

```
>ED CUSTOMERS 500 <r>
TOP
.FS 1000 <r>
TOP
.
```

File 500 under new item-id 1000.

TOP of item 500.

Edit prompt.

Now the item is ready for changes.

5.4 EXITING WITHOUT SAVING

COMMAND	DESCRIPTION
EX	EXIT the Editor without saving the item. If an item-list is active, EDIT the next item on the list, otherwise return to TCL.
EXK	EXIT the Editor without saving the item. KILL any active item-list and return to TCL.

EXITing (EX) an item without saving it is useful to review data non-destructively, or back out of any changes made since your last FILE command.

EDIT is invoked for item 100 on the CUSTOMERS file.

To EXIT without altering anything, enter:

```
>ED CUSTOMERS 100 <r>  
TOP  
.EX <r>  
'100 EXITED
```

>

Back at TCL.

Exiting a single item returns you to TCL. If an item-list is involved, EDIT is automatically invoked on the next item.

```
>ED CUSTOMERS 100 200 300 400 500 <r>  
100  
TOP  
.EX <r>  
200  
TOP  
.EXK <r>
```

>

100 is first.

At the top.

EXIT this item.

Next item is 200.

EXIT and KILL the item-list.

Return to TCL.

Exercise 7

1. Match the command and description.

- | | |
|----------------------|---|
| _____ FSO 600 | a. Flip the work buffers |
| _____ FS | b. FILE to item 600 in the current file and EXIT this item |
| _____ FIK | c. FILE to item 600 in the file BACKUP and EXIT this item |
| _____ FI 600 | d. FILE and overwrite to item 600 in the current file, EXIT this item and KILL any active Item-List |
| _____ F | e. FILE the item and remain in EDIT |
| _____ FS (BACKUP | f. FILE an Overwrite item 600 in the current file and remain in EDIT |
| _____ FS 600 | g. FILE to item 600 in the current file and remain in EDIT |
| _____ FIKO 600 | h. FILE, EXIT and KILL any active Item-List |
| _____ FI (BACKUP 600 | i. FILE this item using the current item-id to the file BACKUP and remain in EDIT |

2. Fill in the appropriate commands.

EDIT the item ATEST in the SAMPLES file and review it.

```
> _____  
TOP  
.  
001 This is the first line.  
002 This is the second line  
003 This is the last line  
EOI 003  
.
```

Enter the command to FILE the current item using item-id BACK.ATEST and EXIT.

```
. _____  
'BACK.ATEST' FILED  
>
```

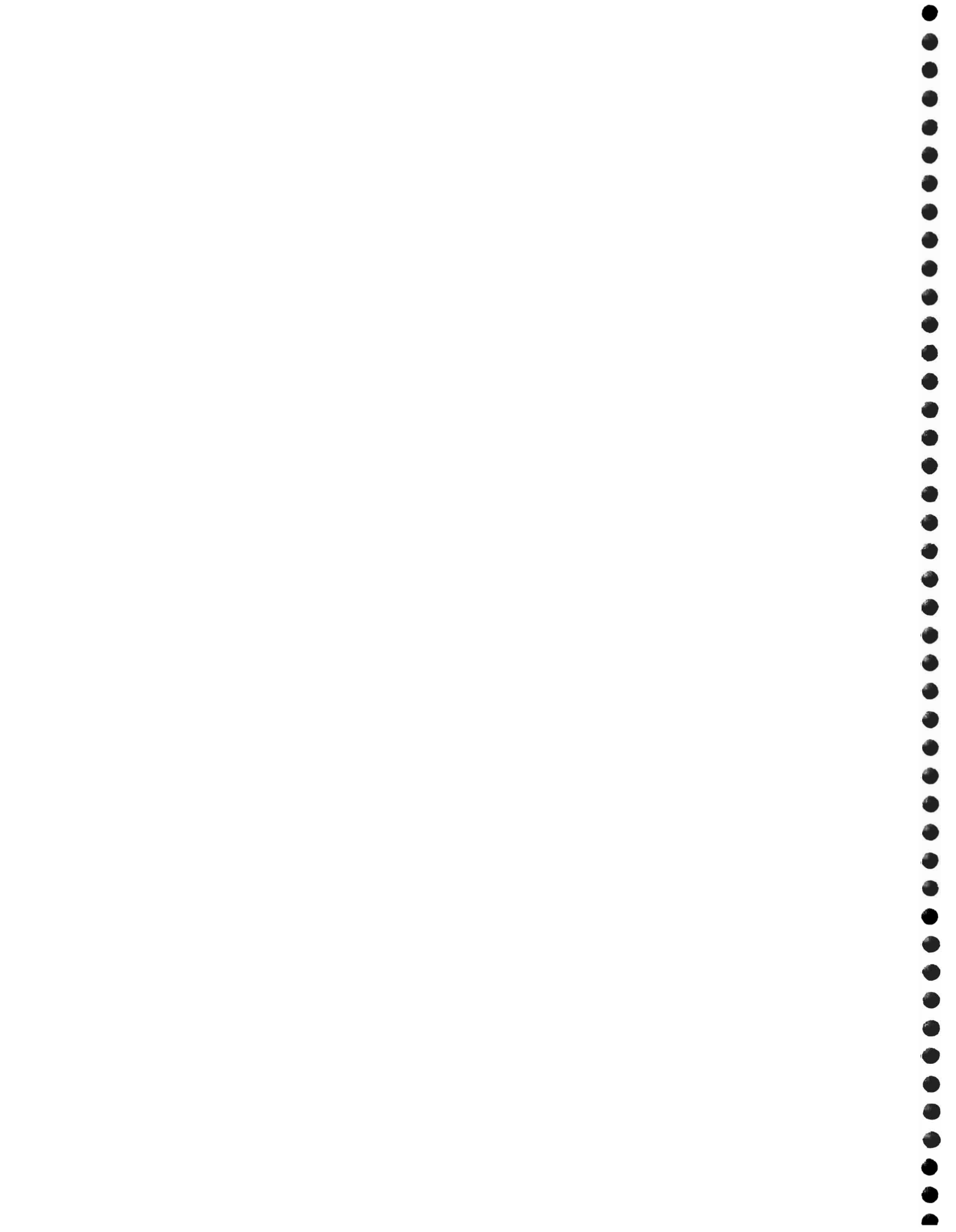
3. Fill in the appropriate commands.

```
>ED SAMPLES 1 2 3 4 5 6 7 <return>
1
TOP
. _____
'1' EXITED
2
TOP
. _____
'2' FILED
3
TOP
. _____
'BACK.3' FILED

TOP
. _____
'3' EXITED

>      Back at TCL
```

NOTES



6. Deleting

6.1 Deleting An Item

An entire item is deleted by invoking the FILE DELETE (FD) command.

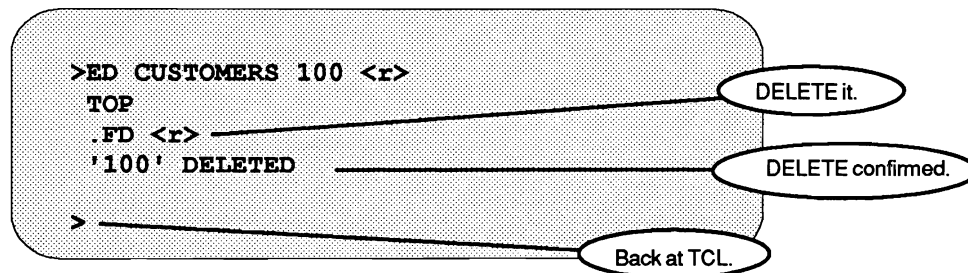
<u>Command</u>	<u>Description</u>
FD	FILE DELETE. Delete the entire item and exit EDIT. If an item-list is active, EDIT the next item. This command can be invoked from any line within the item.
FDK	FILE DELETE and KILL any active item-list. DELETE the entire item and exit to TCL. This command can be invoked from any line within the item.

There is a wise old saying, "FD AND DIE". Remember this.

If you realize that the FD was invoked in haste or by mistake, invoke the TCL command RECOVER-FD. This may be of help in some circumstances. (Some implementations do not have RECOVER-FD, instead they ask "are you sure Y/N ?" when FD is invoked.

DELETE item 100 from the CUSTOMERS file.

Invoke EDIT:



Oops! Don't want to do this.

Invoke RECOVER-FD from TCL.

```
>RECOVER-FD <r>
ITEM NAME:100 <r>

'100' FILED
```

BEWARE! If RECOVER-FD is not performed IMMEDIATELY after the FD, or an item-list is active during EDIT, then the item cannot be recovered.

Instead of invoking RECOVER-FD for your mistake, your implementation might have prompted you as in this example:

```
>ED CUSTOMERS 100 <r>
TOP
.FD
Are you sure? Y <r>
100 deleted
.
```

Invoke EDIT for 4 items in the CUSTOMERS file.

```
>ED CUSTOMERS 100 200 300 400 <r>
100
TOP
.FD <r>
200
TOP
.
```

First item is 100.

At the TOP

Delete it

At the TOP of 200.

DELETE 200, but leave the rest of the list alone.

```
.FDK <r>
```

DELETE and kill the active list.

```
'200' DELETED
```

DELETE confirmed.

```
>_
```

Back at TCL.

6.2 Deleting Lines

Lines can be deleted within an item by invoking the DELETE LINE (DE) command. Line deletes can be performed unconditionally or conditionally.

6.2.1 Unconditional Delete

These commands allow you to delete a single line or a series of lines after and including the current line.

<u>Command</u>	<u>Description</u>
DE	DELETE the current line.
DE number of lines	DELETE the number of lines after and including the current line.
Related Command;	
F	Flip buffers. Reflect the last series of changes to the work item without writing to the file. (See "SAVING YOUR WORK.")

Examples:

- DE Deletes the entire current line of the item.
- DE5 Deletes the current line plus the 4 following.

Invoke EDIT:

```
>ED SAMPLES ATEST <r>
TOP
.L22 <r>
```

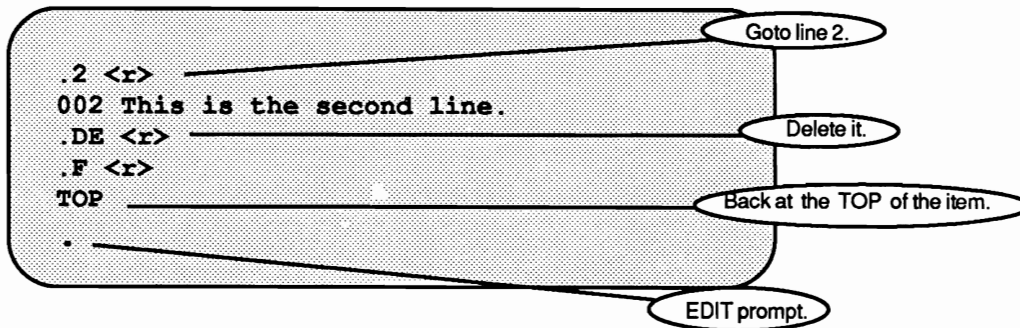
List 22 lines.

Review the item.

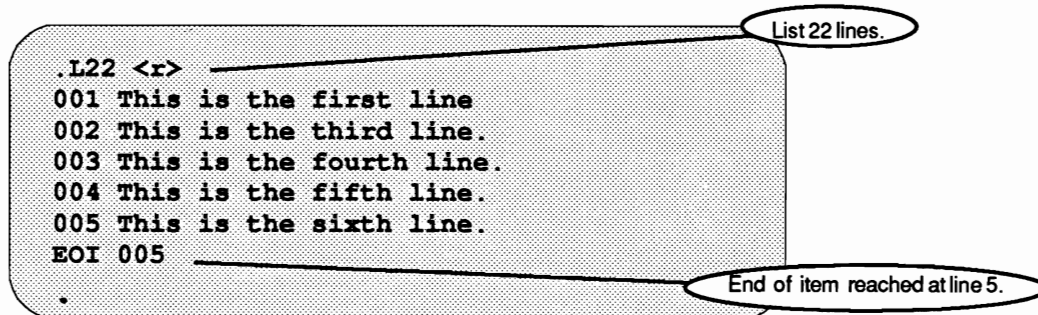
```
001 This is the first line
002 This is the second line.
003 This is the third line.
004 This is the fourth line.
005 This is the fifth line.
006 This is the sixth line.
EOI 006
```

End of item reached at line 6.

DELETE line 2.

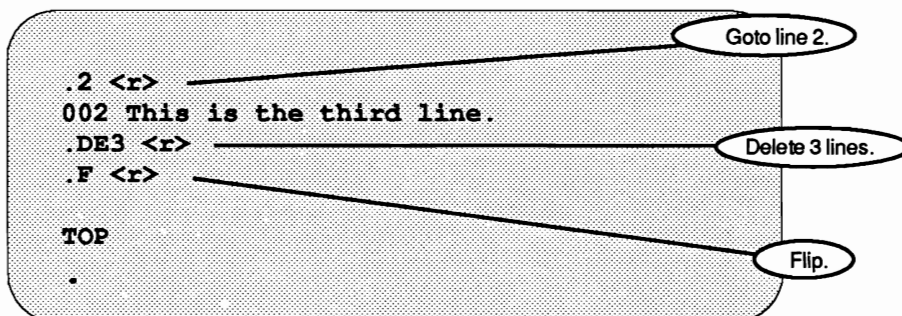


Review the item.



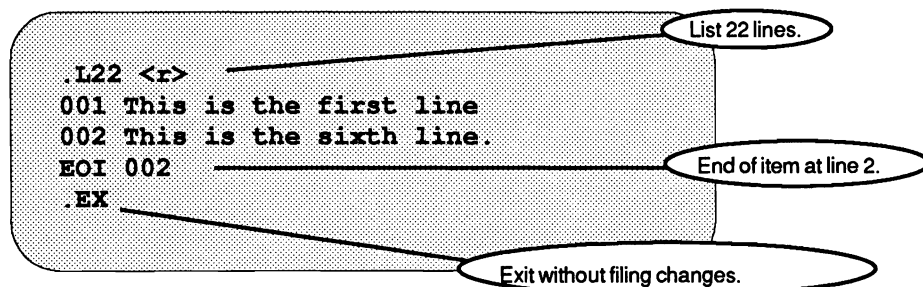
As a result of the flip, The system rennumbers all lines of an item consecutively after a DELETE occurs. This changes the relative position of the remaining attributes. In this case, attribute 3 has shifted to the attribute 2 position, 4 to 3, and so on.

DELETE 3 lines after and including line 2.



Note: Due to the previous (DE) at line 2, what was line 3 is now line 2.

Review the item.



Lines 2 through 4 are deleted. Remember, changes are not permanent until a FILE SAVE command (FS or FI) is invoked.

6.2.2 Conditional Delete

This allows you to delete lines based on a search criterion. Line deletion occurs only if the search string is found in the line.

The search can also be restricted to columnar zones. A columnar zone is a segment of the line bounded by beginning and ending columnar positions.

<u>Command</u>	<u>Description</u>
DEn/string	DELETE multiple lines after and including the current line if the string is found. (n) is the number of lines.
DEn/string/zone	DELETE multiple lines after and including the current line if the string is found within a columnar zone. (n) is the number of lines.
Related Command;	
C	Display columns of the line.

The slash (/) is called a command delimiter. A delimiter separates the pieces of information which make up a command. The delimiter can be any non-numeric character that does not appear in the search string.

Take a look at the item THANKS in the file MEMO.

Invoke EDIT and review the item.

```
>ED MEMO THANKS <r>
```

List 22 lines.

```
TOP
```

```
.L22 <r>
```

```
001 Memo 01/01/87
```

```
002 To: Charlie
```

```
003 From: Me
```

```
004
```

```
005 Charlie,
```

```
006 This is to inform you that your gift was received
```

```
007 and greatly appreciated. A very merry to you too!
```

```
008
```

```
009 Thanks,
```

```
EOI 009
```

End of item at line 9.

Return to the TOP of the item.

```
.T <r>
```

Go to the TOP.

```
TOP
```

DELETE any line if the string "to" appears in it.

```
.DE9/to <r>
```

Delete, within 9 lines, lines which contain the string "to".

```
006 This is to inform you that your gift was received
```

```
007 and greatly appreciated. A very merry to you too!
```

```
EOI 009
```

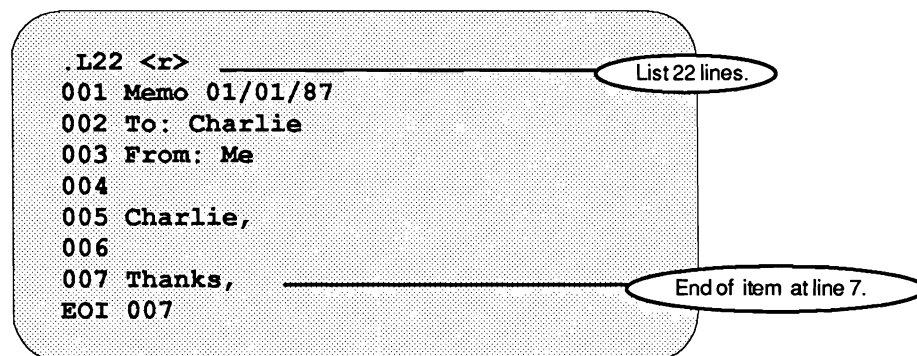
End of item at line 9.

Lines 6 and 7 are deleted. Line 2 remains untouched since "To" does not match "to".

Verify this.



Review the changes.



6.2.3 Deleting Lines Using Zones

You can also specify a search zone as a criterion for deleting a line.

A zone is specified as:

begcolumn-endcolumn

For example:

5-10 specifies a zone between and including columns 5 and 10.

If the beginning column alone is specified, the ending column is the same as the beginning column.

15 specifies a character in column 15.

Examples:

- .DE99/THE/20-60**

DELETE any line within 99 lines after and including the current line, if the string "THE" is found between columns 20 and 60, inclusive.
- .DE99/A/1**

Delete any line within 99 lines after and including the current line, if there is an "A" in column 1.

The "C" command lays out terminal column positions to be used for the zone specification.

```
005 This is the fifth line.  
.C<r>  
      1      2      3...  
123456789012345678901234567890...
```

The word "fifth" begins at column 13 and ends at column 17. It resides in the zone 13-17.

DELETE any line in the item which has the string "Ch" between columns 4 and 10. Use the C command to confirm the column position.

```
.2 <r>  
002 To: Charlie  
.C <r>  
      1      2      3      4  
1234567890123456789012345678901234567890  
.  
.DE9/Ch/4-10
```

Goto line 2.

The "C" in Charley is at position 5.

DELETE any lines which have "Ch" between columns 4 and 10.

```
002 To: Charlie
EOI 007
.F <r>
```

Flip the buffers.

Line 2 is the only one which matches the criterion of capital "C" followed by a lower case "h" (Ch).

Review the item.

```
TOP
.L22 <r>
001 Memo 01/01/87
002 From: Me
003
004 Charlie,
005
006 Thanks,
EOI 006
.
```

List 22 lines.

End of item at line 6.

"Charlie," on Line 4 is not affected because its "Ch" is not within the zone, 4 to 10.

6.2.4 Zone Errors

If an invalid character (non-numeric, outside the column limits) is found in the zone specification, the message

COL#?

is displayed.

Exercise 8

1. The command to delete an item while in EDIT is _____ .
2. The TCL command to recover the most recently deleted item is _____ .
3. True or False.
 - _____ a. FILE DELETE must be invoked at the BOTTOM of the item.
 - _____ b. FDK DELETES the current item and returns to TCL regardless if any Item-List is active.
 - _____ c. RECOVER-FD can be invoked at any time after the item has been deleted.
 - _____ d. FD deletes all the items designated in an active Item-List.
4. Fill in the appropriate commands.

```
>ED SAMPLES ITEM.DELETE.TEST <return>
TOP
: _____
'ITEM.DELETE.TEST' DELETED
```

Now, recover this item.

```
> _____
'ITEM.DELETE.TEST' FILED
>
```
5. Fill in the appropriate comands.

```
>ED SAMPLES A B C D E F <return>
A
TOP
: _____
'A' DELETED
B
TOP
: _____
'B' DELETED
> Back at TCL.
```

Exercise 9

1. Fill in the appropriate command to delete attribute 20.

TOP

. _____
020 This is a test attribute or line as you wish.

. _____
. _____

TOP

.

2. Fill in the commands to delete attributes 15 thru 20, inclusive.

TOP

. _____
015 An attribute of any other name would hold as much data.

. _____
. _____

TOP

.

3. This item has 50 attributes. Fill in the commands to delete them all.

TOP

. _____
. _____

TOP

.

4. The command to display columnar position is _____ .

The following questions use the item DTEST in the SAMPLES file.

ID: DTEST

001 This is a test of the
002 line delete. Deleting lines
003 can be tested conditionally or
004 unconditionally.
005 This is the end of the test item.

5. Fill in the appropriate commands to EDIT this item and DELETE any line with the string "test" in it.

```
> _____  
TOP  
· _____
```

6. Which lines are DELETED? _____

7. Enter the command to delete only those lines which have the letter "s" in columns 1 through 4.

```
TOP  
· _____
```

NOTES



7. Locating

7.1 Locate the First Occurance of a Text String

Any text string can be located within an item by invoking the LOCATE (L) command (not to be confused with the (L)ist command).

LOCATE is invoked as follows:

Command	Description
L/string	LOCATE the first occurrence of the string. LOCATE continues until the string is found or the bottom of the item is reached.
Related Command:	
A	Again. Repeats the last LOCATE command.

The slash (/) is called a command delimiter. The presence of the delimiter after the L indicates that this is a LOCATE and not a LIST command. A delimiter separates the pieces of information which make up a command. The delimiter can be any non-numeric character which does not appear in the search string.

LOCATE begins from the line just following the current line.
The line pointer is incremented to the position of the last line in the search.

Examples:

L/fifth	Search for the first occurrence of "fifth"
L/this	Search for the first occurrence of the string "this".
L*/	Search for the first occurrence of '/'. Notice the delimiter is an '*'.

Invoke EDIT and review the item.

```
>ED SAMPLES ATEST <r>
```

```
TOP
```

```
.L22 <r>
```

List 22 lines.

```
001 This is the first line.
```

```
002 This is the second line.
```

```
003 This is the third line.
```

```
004 This is the fourth line.
```

```
005 This is the fifth line.
```

```
006 This is the sixth line.
```

```
EOI 006
```

End of item at line 6.

LOCATE the first occurrence of the string "is".

```
.L/is <r>
```

Line pointer is at line 001.

```
001 This is the first line
```

```
.
```

The first occurrence of "is" is found on line 001.

Do it again.

```
.A <r>
```

LOCATE again.

```
002 This is the second line.
```

```
.
```

The next occurrence is found on line 2.

The (A) command will locate the string 'is' until a new (L)OCATE is invoked.

```
.A <r>
```

Locate the next occurrence of 'is'.

```
003 This is the third line
```

```
.L/fourth <r>
```

Locate the text string 'fourth'.

```
004 This is the fourth line]
```

```
.A <r>
```

Locate 'fourth' again.

```
EOI 006
```

'fourth' not found again.
End of Item at line 6.

7.2 Locating the First Occurance In Multiple Lines

Ln/string	LOCATE each occurrence of the string on multiple lines, following the current line. (n) is the number of lines to search.
Related Command:	
A	Again. Repeats the last LOCATE command.

Show every occurrence of "is" in ATEST.

```
.T <r>
TOP
.L99/is <r>
001 This is the first line.
002 This is the second line.
003 This is the third line.
004 This is the fourth line.
005 This is the fifth line.
006 This is the sixth line.
EOI 006
.
```

Go to the TOP.

LOCATE every occurrence of "is" within the next 99 lines. Since the item has only 6 lines, the search will terminate at the bottom of the item.

End of item at line 6.

Every line in this item is displayed since each line contains the search string 'is.' The current line is now 6.

```
.T
TOP
.L99/fourth <r>
004 This is the fourth line
EOI 006
.
```

Return to the top of the item.

Locate the first occurrence of 'fourth' over the next 99 lines.

Line 4 is the only line that 'fourth' is found.

7.3 Locate Within Zones

The search can also be restricted to columnar zones. A columnar zone is a segment of the line bounded by beginning and ending columnar positions.

<u>Command</u>	<u>Description</u>
L/string/zone	LOCATE the first occurrence of the string within the specified columnar zone.
Ln/string/zone	LOCATE each occurrence of the string, within column boundaries, on multiple lines, following the current line. (n) is the number of lines to search.
Related Command;	
A	LOCATE again. Repeat the last LOCATE command.
C	Display columns of the line.

The line pointer is incremented to the position of the last line of the search.

The delimiter (/) can be any non-numeric character that does not appear in the search string.

The zone is specified as a range of columnar positions:

begcolumn-endcolumn

5-10 specifies a zone between and including columns 5 and 10.

Example:

.L/THE/20-60	LOCATE the line with the first occurrence of "THE" if it appears between columns 20 and 60, inclusive.
.L99/A/1	LOCATE within the next 99 lines all occurrences of an "A" in column 1.

The C command causes a display of terminal column positions to be used for any zone specification.

```
.C <r>  
      1      2      3...  
123456789012345678901234567890..
```

Invoke EDIT and review the item.

```
>ED MEMO THANKS <r>  
TOP  
.L22 <r>  
001 Memo 01/01/87  
002 To: Charlie  
003 From: Me  
004  
005 Charlie,  
006 This is to inform you that your gift was received  
007 and greatly appreciated. A very merry to you too!  
008  
009 Thanks,  
EOI 009
```

List 22 lines.

End of item at line 9.

Return to the TOP of the item.

```
.T <r>
```

```
TOP  
.
```

Go to the TOP.

Try locating a text string using a zone specification. Search for the first occurrence of the string "Ch" between columns 4 and 10.

Use the (C) command to confirm the column positions you want to search.

```
.2 <r>
002 To: Charlie
.C <r>
```

1 2 3 4

1234567890123456789012345678901234567890

Goto line 2.

The "C" command shows that "C" in Charlie is at columnar position 5.

```
.L/Ch/4-10
002 To: Charlie
.
```

Locate the line with the first occurrence of "Ch" between columns 4 and 10.

Line 2 has the first occurrence of Ch within the specified search zone.

Now, LOCATE all occurrences of the string "to" between columns 1 and 15.

Return to the TOP of the item.

```
.T <r>
TOP
.L99/to/1-15 <r>
006 This is to inform you that your gift was received
EOI 006
.
```

Goto the TOP.

LOCATE within the next 99 lines, all occurrences of "to" if it appears between columns 1 and 15, inclusive.

End of the Item.

The search continues until 99 lines are searched or the bottom of the item is reached. Searches in Release "R83" implementations are case sensitive. Line 002, "To: Charlie" was not included because "to" does not match "To". This may not be true in your implementation. Try this and see what happens.

7.4 Wildcard Searches

There may be situations where you need to LOCATE a string that has a series of "don't care" characters imbedded. For example, you may wish to search for the first occurrence of a 5 character string which begins with an "A" and ends with an "N".

This is designated by the command:

L/A^^^N LOCATE the first occurrence of "A", 3 "don't care" characters, and an "N".
In this case, any line with "AGAIN," "AROUND," etc. will satisfy the LOCATE request.

The caret "^" is a "wildcard" search character. Use it when you only care if a character is there or not, but you don't care what the value is.

The caret "^" is generated by a <shift>6 from most keyboards. Do not confuse this with the <ctl><Shift>6 (attribute mark).

WILDCARD searches are very useful when trying to LOCATE <ctl> characters in a line. These <ctl> characters display as periods "." and visually cannot be distinguished from actual periods.

If you need to LOCATE the caret "^" as an actual part of the text, the wildcard search must be turned off.

The command to turn off wildcard searches is;

<u>Command</u>	<u>Description</u>
^	Toggle (turn off and on) wildcard search. This is the up-arrow.

Upon invocation of EDIT, the wildcard search is always active by default.

EDIT item ANOTHERTEST in the file SAMPLES.

Invoke EDIT and review the item.

```
>ED SAMPLES ANOTHERTEST<r>
```

```
TOP
```

```
.L22
```

```
001 This is an example of a wildcard  
002 search. Wildcards are designated  
003 by the up-arrow (^). This should  
004 wind up the discussion of  
005 wildcards.
```

```
EOI 005
```

```
.
```

List 22 lines.

End of item at line 5.

Go back to the TOP and search for any occurrence of a "w" followed by 2 characters and a "d".

```
.T<r>
```

```
TOP
```

```
.L99/w^^d<r>
```

```
001 This is an example of a wildcard  
004 wind up the discussion of  
005 wildcards
```

```
EOI 005
```

```
.
```

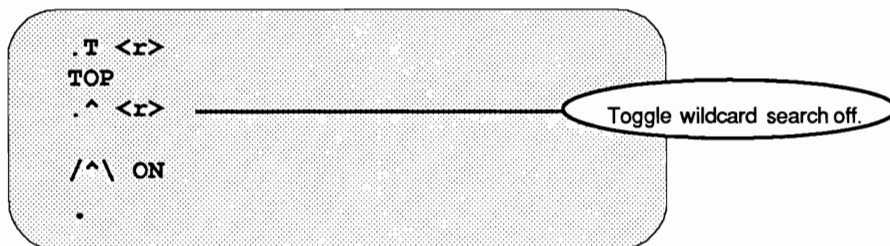
Goto the TOP.

LOCATE within the next 99 lines each occurrence of a "w", any character, and a "d".

End of item at line 5.

Lines 1, 4 and 5 meet the criteria. Notice that lines 1 and 5 matched on the word "wildcard," while line 4 matched on "wind."

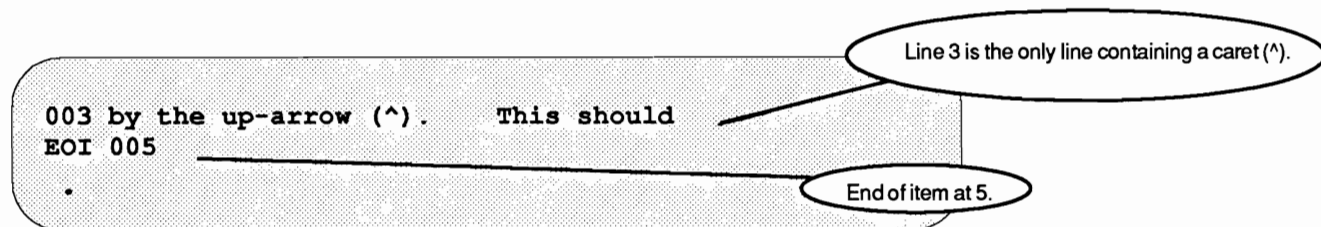
Now, return to the TOP and search for an actual caret "^".



This message means that the caret "^" is now "ON" as a character and not a wildcard.

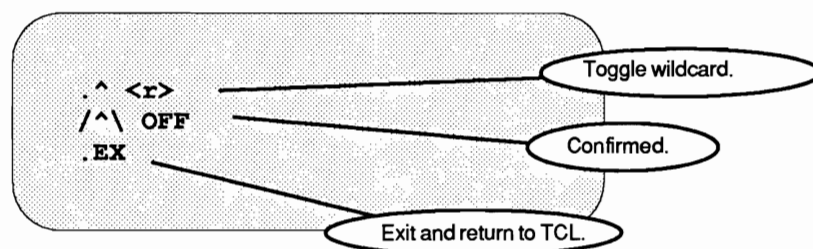
.L99/^<return>

LOCATE within the next 99 lines each occurrence of the string "(^)".



The string "(^)" is LOCATED on line 3.

Now, toggle the wildcard search back on by turning the "^" "OFF" as a character.



Exercise 10

1. Fill in the command to LOCATE the first occurrence of "HELLO".

TOP

. _____

What is the command to repeat the last LOCATE?

. _____

2. Fill in the command to LOCATE all occurrences of "123" within an item of 120 attributes.

TOP

. _____

What is the current line at the end of the LOCATE?

3. Fill in the command to LOCATE "ABC" in the ext 20 lines.

. _____

4. For the same item with 120 lines, fill in the command to LOCATE the first occurrence of the letter "X" in column 1.

TOP

. _____

5. Fill in the command to LOCATE all occurrences of "XYZ" in columns 1 through 10.

. _____

6. In an item that has 200 lines, fill in the commands to LOCATE all occurrences of the character up-arrow (^).

TOP

. _____

/^\ ON

. _____

Exercise 10 - Page 2

7. Match the command and description.

_____ L/WE

_____ L99/XYZ/3-10

_____ A

_____ L/L/20

_____ L*/

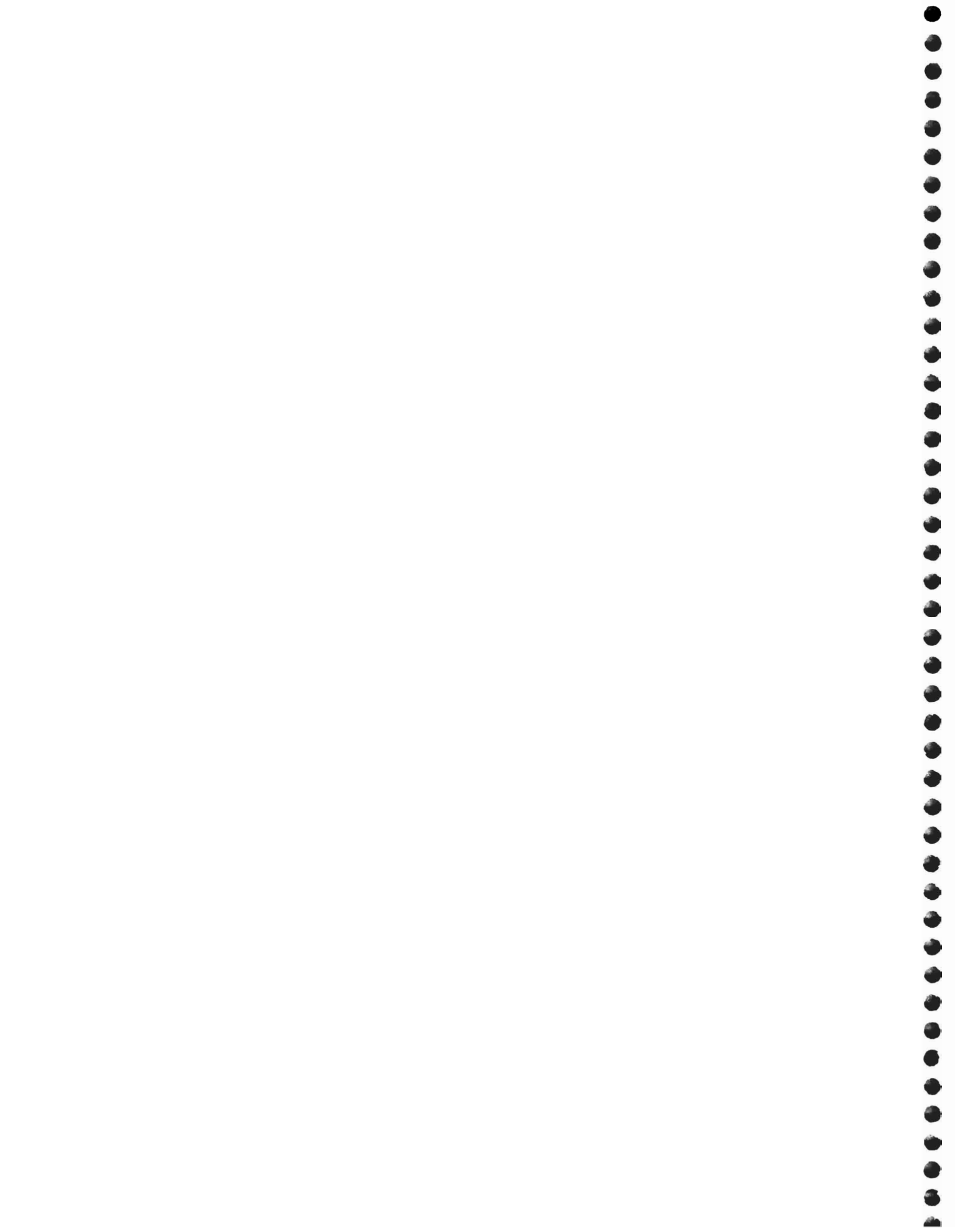
_____ L99/*

_____ L/^A^/1-3

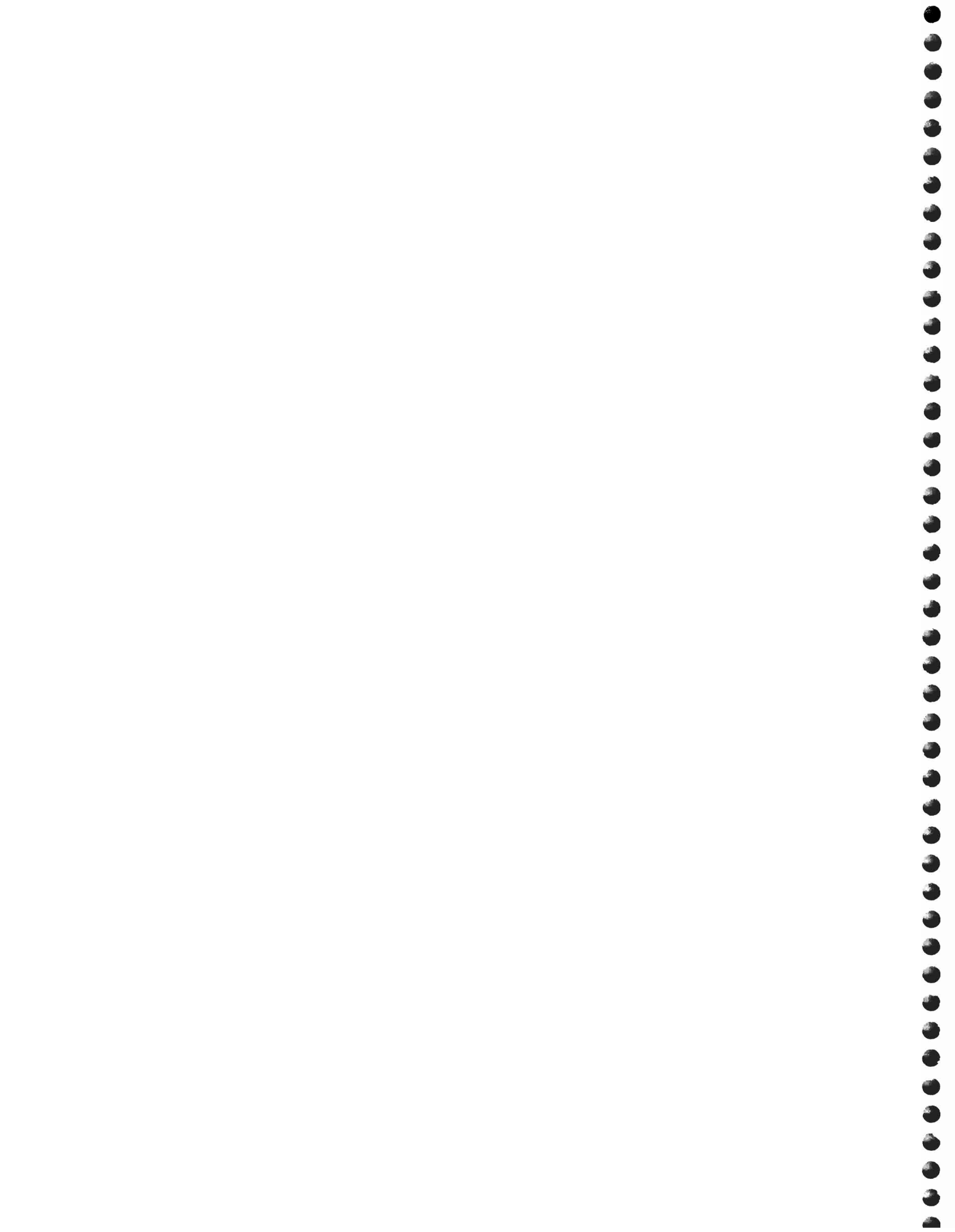
_____ ^

_____ L/AN^

- a. Toggle WILDCARD search.
- b. LOCATE the first occurrence of a slash (/).
- c. LOCATE the first occurrence of "AN" followed by any character.
- d. LOCATE the first occurrence of any character, followed by an "A", followed by any character in columnar positions 1 through 3.
- e. LOCATE all occurrences of an asterisk (*) in the next 99 lines.
- f. Repeat the last LOCATE.
- g. LOCATE all occurrences of "XYZ" between columns 3 and 10 in the next 99 lines.
- h. LOCATE the first occurrence of "WE".
- i. LOCATE the first occurrence of "L" in column 20.



NOTES



8. Replacing

The process of revising existing lines is invoked by the command (R) REPLACE LINES.

REPLACE can be performed on either an entire line or portions of a line.

In addition, REPLACE allows you to change a single line or multiple lines after and including the current line.

<u>Command</u>	<u>Description</u>
R	REPLACE the entire current line.
R number of lines	REPLACE in entirety the number of lines after and including the current line.
Related Command:	
F	Flip buffers. Reflect the last series of changes to the work item without writing to the file. (See "Saving Your Work.")

Examples:

- R** REPLACE the entire current line of the item.
- R5** REPLACE, in entirety, 5 lines including the current line in and the 4 following.

When command R is invoked, EDIT prompts for input of the entire line. As in INSERT, the prompt character is the plus sign (+).

Invoke EDIT and review the item:

```
>ED SAMPLES ATEST <r>
```

```
TOP
```

```
.L22 <r>
```

List 22 lines.

```
001 This is the first line
002 This is the second line.
003 This is the third line.
004 This is the fourth line.
005 This is the fifth line.
006 This is the sixth line.
```

```
EOI 006
```

End of item reached at line 6.

REPLACE line 6.

```
.6 <r>
006 This is the sixth line.
.R <r>
006+ _
```

EDIT prompts with the line number (006) and the plus sign to indicate input is now expected.

Now, enter the following:

```
006+ This is the last line. <r>
.
```

Flip the buffer to reflect the change.

```
.F <r>
TOP
.
```

Review the item.

```
.L22 <r>
001 This is the first line.
002 This is the second line.
003 This is the third line.
004 This is the fourth line.
005 This is the fifth line.
006 This is the last line.
EOI 006
.
```

Next, REPLACE 4 lines after and including line 2.

```
.2 <r>
002 This is the second line.

.R4 <r>
002+ This is a line. <r>
003+ This is another line. <r>
004+ This is also a line. <r>
005+ This is the 4th line of the replace. <r>
.
```

Flip the buffer to reflect the changes.

```
.F <r>
TOP
.
```

Flip.

Review the item.

```
.L22 <r>
001 This is the first line
002 This is a line.
003 This is another line.
004 This is also a line.
005 This is the fourth line of the replace.
006 This is the last line.
EOI 006
.EX <r>
```

List 22 lines.

End of item at line 6.

Exit without filing changes.

Lines 2 through 5 are REPLACED. Remember, changes are not permanent until a FILE SAVE command (FS or FI) is invoked.

8.2 Replacing Portions Of A Line

This allows you to search for an existing text string and REPLACE it with a new text string. Replacement occurs only if the search string is found.

The search can also be restricted to columnar zones. A columnar zone is a segment of the line bounded by beginning and ending column positions.

Command	Description
R/oldstring/newstring/	REPLACE the first occurrence of oldstring with newstring on the current line.
Rn/oldstring/newstring/	REPLACE within the specified number of lines (n) after and including the current line, the first occurrence per line of oldstring with newstring.
Related Command	
F	Flip buffers. Reflect the last series of changes to the work item without writing to the file. (See "SAVING YOUR WORK.")

The slash (/) is called a command delimiter. A delimiter separates the pieces of information which make up a command. The delimiter can be any non-numeric character that does not appear in the search string.

Examples:

.R99/THE/THOSE REPLACE, within the next 99 lines, the first occurrence of "THE" with "THOSE" on each line.

.R99/FIRST/LAST REPLACE, within the next 99 lines, the first occurrence of "FIRST" with "LAST" on each line.

.R!/!! REPLACE, on the current line, the first occurrence of a slash '/' with null. Notice the delimiter is the exclamation point '!'.

The 'C' command displays terminal column positions to be used for the range, as in this example:

```
005 This is the fifth line.
```

```
.C <r>
```

```
          1          2          3...
123456789012345678901234567890...
```

The word "fifth" begins at column 13 and ends at column 17. It resides in the zone 13-17.

EDIT the item 'THANKS' in the file 'MEMO' and review the item.

```
>ED MEMO THANKS <r>
```

```
TOP
```

```
.L22 <r>
```

```
001 Memo 01/01/87
```

```
002 To: Charlie
```

```
003 From: Me
```

```
004
```

```
005 Charlie,
```

```
006 This is to inform you that your gift was received
```

```
007 and greatly appreciated. A very merry to you too!
```

```
008
```

```
009 Thanks,
```

```
EOI 009
```

List 22 lines.

End of item at line 9.

Return to the TOP of the item.

```
.T <r>  
  
TOP  
.
```

Go to the TOP.

REPLACE on every line in the item, the first occurrence of "to" with "two".

```
.R9/to/two <r>  
006 This is two inform you that your gift was received  
007 and greatly appreciated. A very merry two you too!  
EOI 009  
.
```

REPLACE, within 9 lines, the first occurrence of "to" with "two".

End of item at line 9.

Lines 6 and 7 are REPLACEd. Line 2 remains untouched since "To" does not match "to". The "too!" on line 7 is the second occurrence of "to" on that line and is therefore unaffected.

Verify.

```
.F <r>  
  
TOP  
.
```

Flip the buffers so the changes will reflect in the work item.

Review the changes.

```
.L22<r>
001 Memo 01/01/87
002 To: Charlie
003 From: Me
004
005 Charlie,
006 This is two inform you that your gift was received
007 and greatly appreciated. A very merry two you too!
008
009 Thanks,
EOI 009
```

List 22 lines.

End of item at line 9.

8.3 Replacing Using Zones

R/oldstring/newstring/zone	REPLACE, within the zone limits, the first occurrence of oldstring with newstring on the current line.
Rn/oldstring/newstring/zone	REPLACE, within the zone limits, the first occurrence of oldstring with newstring on multiple lines. (n) is the number of lines to search.
Related Command;	
C	Display columns of the line.

A zone is specified as: **begcolumn-endcolumn**

If the begcolumn is the only zone specification, the endcolumn is the same as the begcolumn.

For example: **5-10 specifies a zone between and including columns 5 and 10.**
5 specifies the zone as column 5 only.

Examples:

.R99/A/X/1	REPLACE, within the next 99 lines, each occurrence of "A" in column 1 with "X".
.R99/S/ES/3-7	REPLACE, within the next 99 lines, each occurrence of "S" in columns 3 through 7 with "ES."

Replace on every line the first occurrence of "Ch" between columns 4 and 10 with an "F".

Use the C command to confirm the column positions to search.

```
.2 <r>
002 To: Charlie
.C <r>
      1      2      3      4
1234567890123456789012345678901234567890
.
.R9/Ch/F/4-10
002 To: Farlie
EOI 009
.F <r>
```

Goto line 2.

The "C" in Charley is at position 5.

REPLACE within 9 lines, the first occurrence of "Ch" between columns 4 and 10 with an "F".

End of item at line 9.

Flip the buffers.

Line 2 is the only one which matches the criterion.

Review the item.

```
TOP
.L22 <r>
001 Memo 01/01/87
002 To: Farlie
003 From: Me
004
005 Charlie,
006 This is two inform you that your gift was received
007 and greatly appreciated. A very merry two you too!
008
009 Thanks,
EOI 009
```

List 22 lines.

End of item at line 9.

"Charlie," on Line 5 is not affected because its "Ch" is not within the zone, 4 to 10.

```
.T <r>
TOP
.
```

Return to the top of the item.

8.4 Universal Replace

Each of the above REPLACE (R) commands replaces only the FIRST occurrence of a string on a given line. If a string appears more than once on the same line, the command must be invoked repeatedly to change each occurrence. The UNIVERSAL REPLACE command eliminates the need to do this.

The UNIVERSAL REPLACE is the RU command.

<u>Command</u>	<u>Description</u>
RU/oldstring/newstring/	REPLACE all occurrences of oldstring with newstring on the current line.
RUn/oldstring/newstring/	REPLACE all occurrences of oldstring with newstring on multiple lines after and including the current line. (n) is the number of lines.

Example: **.RU999/X/+** REPLACE every occurrence of "X" with "+" within 999 lines, starting with the current line. This is effectively a global search and replace.

The UNIVERSAL REPLACE replaces EVERY occurrence of the string on a line, NOT just the first occurrence.

Now try this. REPLACE every occurrence of a lower case "t" with an asterisk "*".

Review the item.

```
TOP
.L22 <r>
001 Memo 01/01/87
002 To: Farlie
003 From: Me
004
005 Charlie,
006 This is two inform you that your gift was received
007 and greatly appreciated. A very merry two you too!
008
009 Thanks,
EOI 009
```

List 22 lines.

End of item at line 9.

Notice that all the previous changes are still present.

```
.RU99/t/* <r>
006 This is *o inform you *ha* your gif* was received
007 and grea*ly apprecia*ed. A very merry *o you *oo!
EOI 009
```

Replace every occurrence of "t" with "" within 99 lines.

End of item at line 9.

Notice that the upper case T's are not affected.

8.5 Replacing At The Beginning Of A Line (Prefixing)

Sometimes, it is necessary to "insert" text at the beginning of a line.

For example: Insert the word "Many" at the beginning of line 9 so it will say "Many Thanks".

```
.9 <r>
009 Thanks,
.R//Many<space>/ <r>
```

Goto line 9.

Two consecutive delimiters (//) indicate a null. This says to replace the first null with "Many" followed by a space. Notice that a space is imbedded after the word "Many" so that it will be separate from "Thanks,". The last delimiter is necessary to indicate the trailing blank, otherwise it would be stripped.

```
009 Many Thanks,
.F <r>
TOP
```

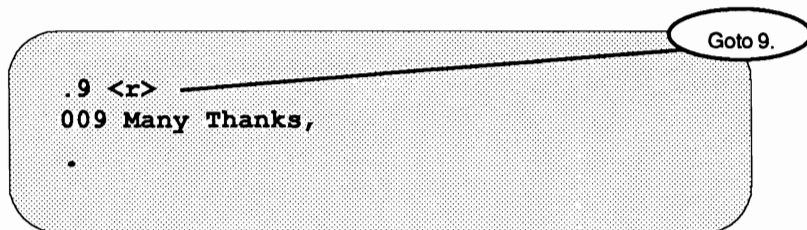
Flip the buffers.

8.6 Replacing At The End Of A Line (Appending)

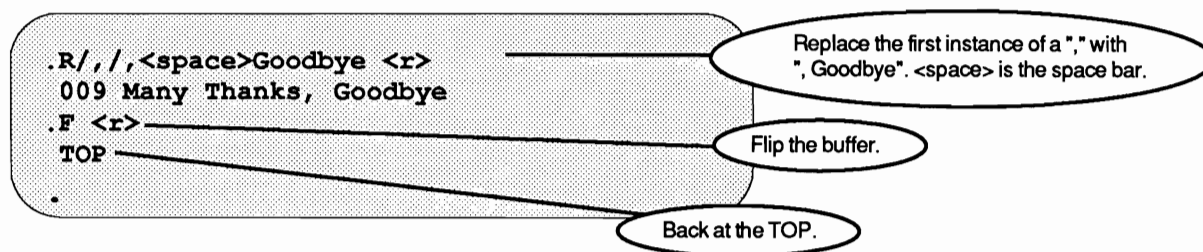
Sometimes it is necessary to add text to the end of an existing line. Two of the three methods to accomplish this are discussed here. The third method is called the Append Line (AL) command, and can only be used if your implementation allows it. Check your manuals for the exact syntax.

8.6.1 Method 1

On line 009, add the word "Goodbye".



Find a unique text string at the end of the line. In this case, a comma (,) ends the line.



Flip the buffer.

Back at the TOP.

8.6.2 Method 2

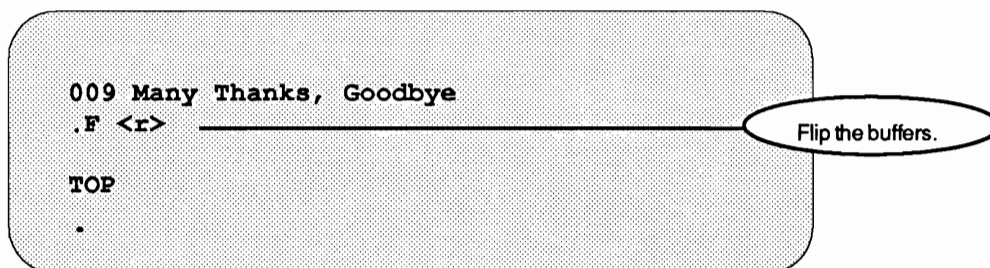
Another method can be used if the line is long, and there are no obviously unique text strings.

The End of Line can be thought of as a long string of blanks even though the system does not actually store the blanks. Replace a string of blanks with the text you wish to add. Be careful to make sure you specify a string of blanks longer than what is already imbedded in the line.

Use the C command to display column positions and determine the number of repeated characters within a line.

.R/<space><space><space><space>/<space>Goodbye

Replace the first occurrence of 4 blanks with "Goodbye". Notice that the space is imbedded at the beginning of the newstring "Goodbye".



8.7 Truncating A Line

A line can be truncated (shortened) by ending the REPLACE string with an attribute mark (<ctl><shift>6).

Truncate line 9 from the previous example. Truncate the line immediately following the comma (,).

REPLACE the first comma "," with a "," followed by an attribute mark. The attribute mark appears on the terminal as an up-arrow (^). (R/,/,^) This will delete everything after the ",".

```
009 Many Thanks,
```

```
.
```

```
.F
```

```
TOP
```

```
.9
```

```
009 Many Thanks,
```

Flip the buffer.

Goto line 9.

Now, let's truncate the line after 'many':

```
.R/Y/Y<ctl><shift>6 <r>
```

```
009
```

```
.EX <r>
```

Exit without filing the changes.

Depending on which implementation you are using, this command will work differently. For example, on Ultimate this will break the line into 2 attributes rather than truncating.

8.8 Wildcard Replaces

There may be situations where you need to replace a string that has a series of "don't care" characters imbedded. For example, you may wish to search for the first occurrence of a 5 character string which begins with an "A" and ends with an "N" and replace it with a NULL.

This is designated by the command,

```
R/A^^^N//
```

On the current line, REPLACE the first occurrence of "A", 3 "don't care" characters, followed by an "N" with NULL In this case, any line with "AGAIN", "AROUND", etc., will satisfy the REPLACE request.

The up-arrow (^) is a "wildcard" search character. Use it when you want to know if a character is there or not, but you don't care what the value is.

The up-arrow (^) is generated by a <shift>6 from most keyboards. Do not confuse this with the <ctl><Shift>6 (attribute mark).

WILDCARD searches are very useful when trying to REPLACE <ctl> characters in a line. These <ctl> characters display as periods (.) and visually cannot be distinguished from actual periods.

If you need to locate the up-arrow (^) as an actual part of the text, the wildcard search must be turned off.

The command to turn off wildcard searches is:

<u>Command</u>	<u>Description</u>
^	Toggle (turn off and on) wildcard search. This is the up-arrow.

The wildcard REPLACE is active upon invocation of EDIT.

EDIT item ANOTHERTEST in the file SAMPLES.

Invoke EDIT and review the item.

```
>ED SAMPLES ANOTHERTEST <r>
TOP
.L22
001 This is an example of a wildcard
002 search. Wildcards are designated
003 by the up-arrow (^). This should
004 wind up the discussion of
005 wildcards.
EOI 005
.
```

List 22 lines.

End of item at line 5.

Go back to the TOP and REPLACE all occurrences of a "w" followed by 2 characters and a "d" by NULL.

```
.T <r>  
TOP  
.RU99/w^^d// <r>
```

Goto the TOP.

REPLACE within 99 lines, each occurrence of a "w", any character, and a "d" by a NULL.

```
001 This is an example of a card  
004 up the discussion of  
005 cards  
EOI 005  
.
```

End of item at line 5.

Lines 1, 4 and 5 meet the criteria. Notice that lines 1 and 5 matched on the word "wildcard", while line 4 matched on "wind".

Now, return to the TOP and REPLACE all occurrences of the string " (^)" with the word "up-arrow".

```
.T <r>  
TOP  
.^ <r>  
/^ \ ON  
.
```

Toggle wildcard search off.

This message means that the "^" is now "ON" as a character and not a wildcard.

```
.RU99/(^)/up-arrow<return>
```

REPLACE within 99 lines each occurrence of the string " (^)" with the word "up-arrow".

```
003 by the up-arrow up-arrow. This should  
EOI 005
```

End of item at 5.

The string " (^)" up-arrow is located on line 3.

Now, toggle the wildcard back on.

```
. ^ <r>  
/^ \ OFF
```

Toggle wildcard.

Confirmed

The "^" is now "OFF" as a character as is used for wildcard searches once again.

Caution: The Wildcard Replace can be used to replace every character in an item. Try this if you wish, but don't file the changes.

```
. F <r>  
TOP
```

```
. RU999/^/X
```

Flip and go to the top

Replace universally over the next 999 lines every character with an "X"

Note what happens. (Olly North should have known about this!)

Exercise 11

1. Using item ATEST in the file SAMPLES:

ID: ATEST

001 This is the first line.
002 This is the second line.
003 This is the third line.
004 This is the fourth line.
005 This is the fifth line.
006 This is the sixth line.

Fill in the commands to replace line 2 entirely by the text string "This is replaced line 2".

```
> _____  
TOP  
.  
002 This is the second line.  
.  
002+ _____  
.  
TOP  
.
```

2. Fill in the commands to REPLACE all of lines 3, 4, and 5 with "These are the replaced lines".

```
TOP  
.  
003 This is the third line.  
.  
003+ _____  
004+ _____  
005+ _____  
.  
TOP  
.
```

3. On line 6, REPLACE "This is" with "That was".

```
TOP  
.  
006 This is the sixth line.  
.  
006 That was the sixth line.  
.  
TOP  
.
```

4. Enter the command to REPLACE the first occurrence on every line of "This" with "That".

TOP

. _____

What is the command to do this to every occurrence?

TOP

. _____

5. Fill in the command to REPLACE all occurrences of an "A" in column 10 with an asterisk (*).

TOP

. _____

6. Fill in the appropriate commands to put the word START before the first word of the first line.

TOP

. _____

001 This is the first line.

. _____

001 START This is the first line.

. _____

TOP

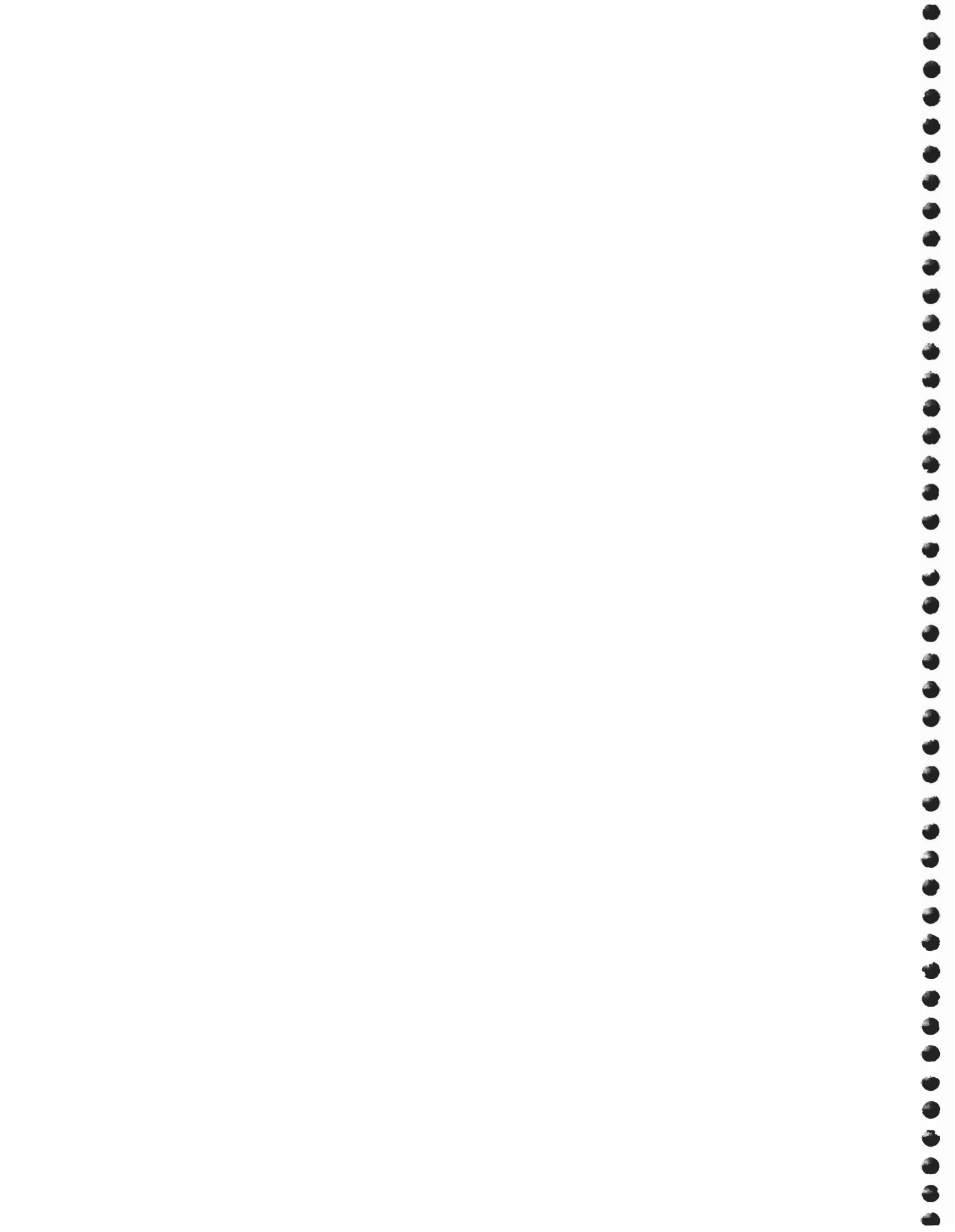
.

Exercise 11 - page 3

7. Match the command with its description.

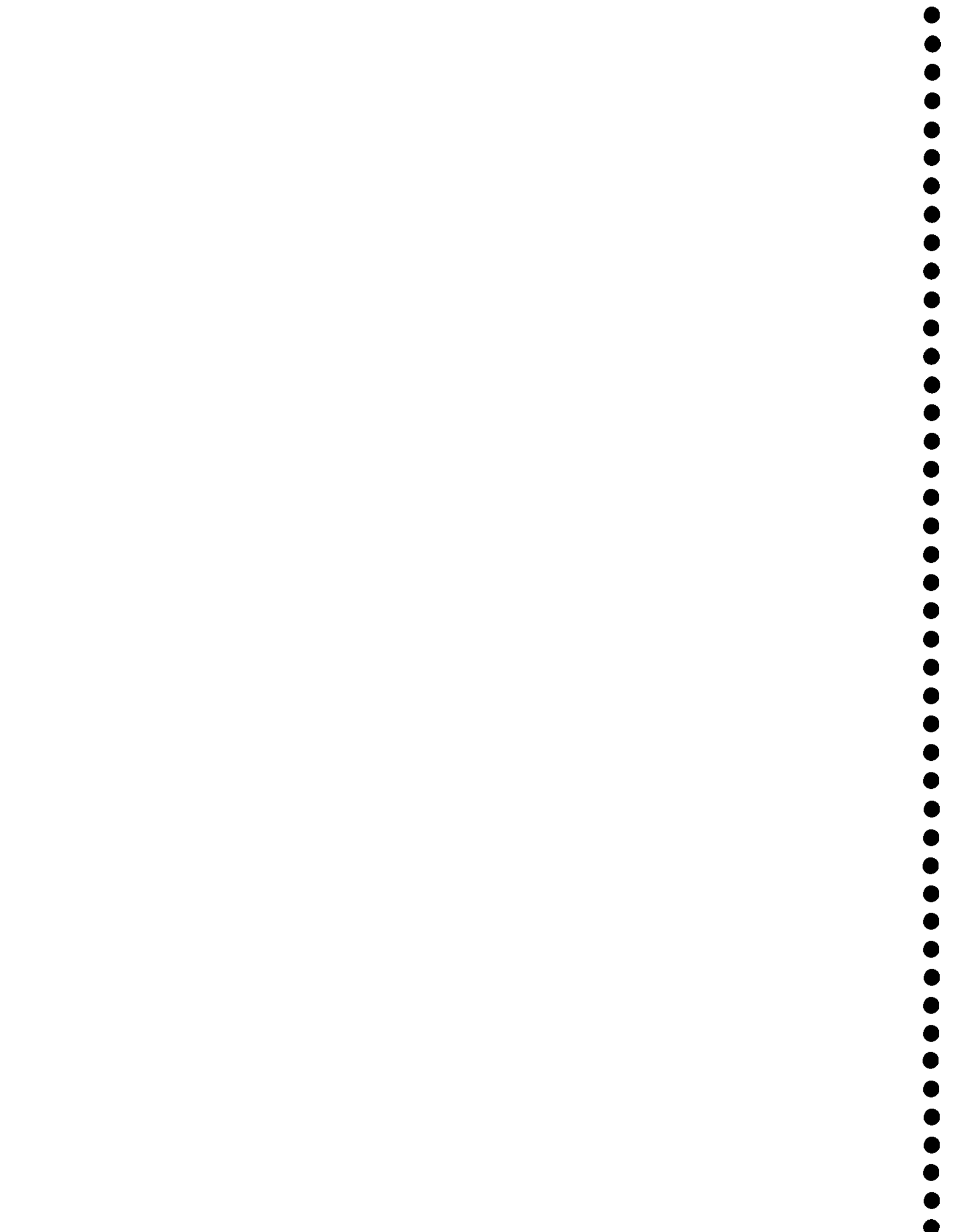
Assume these commands are invoked in an item with 50 attributes.

- | | |
|---------------------|---|
| _____ R | a. REPLACE 5 lines. |
| _____ R/ABC/CBA/3-6 | b. Toggle WILDCARD replace. |
| _____ R/123/321 | c. REPLACE every occurrence of an "X" in column 10 with an "A". |
| _____ R*/*- | d. REPLACE all occurrences of asterisk (*) with an exclamation (!). |
| _____ R99/A/L | e. REPLACE the first occurrence of any character followed by an "A" between columns 1 and 3, with an "A". |
| _____ RU99/*/! | f. REPLACE in the current line the first occurrence of "ABC" between columns 3 and 6, with "CBA". |
| _____ R99/X/A/10 | g. REPLACE the entire current line. |
| _____ R5 | h. REPLACE in the current line the first occurrence of slash (/) with a dash (-). |
| _____ R99/^A/A/1-3 | i. REPLACE on every line the first occurrence of "A" with an "L". |
| _____ ^ | j. REPLACE on the current line the first occurrence of "123" with "321". |





NOTES



9. Merging

9.1 Merging Lines

MERGING LINES (ME) is the process of inserting a copy of a portion of an item following the current line position of the current item.

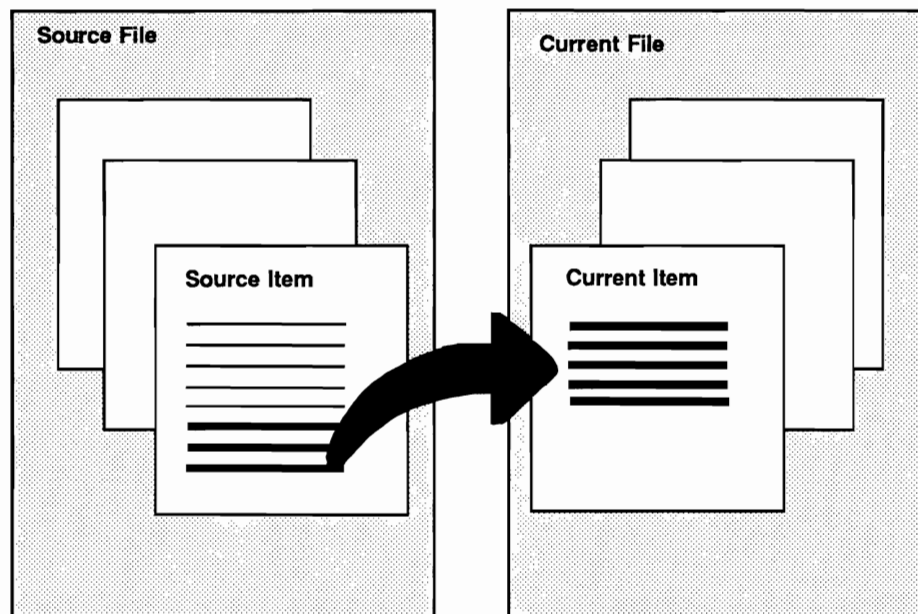
Lines can be MERGEEd from the same item or a different item within the current file or from a different file. The destination of a MERGE is always the current item in the current file.

The item presently being EDITed is referred to as the current item.

The file from which the current Item is retrieved is the current file.

The item from which lines are retrieved for the MERGE is referred to as the source item.

The file from which the source item is retrieved is referred to as the source file.



9.2 Merging From The Same File

The source file is the same as the current file.

<u>Command</u>	<u>Description</u>
ME <i>n</i> / <i>item-id</i> / <i>s</i>	MERGE a number of lines (<i>n</i>) from the source item addressed by <i>item-id</i> starting at line (<i>s</i>) in the source item.

All MERGE commands INSERT lines immediately following the current line.

The slash (/) is called a command delimiter. A delimiter separates the pieces of information which make up a command. The delimiter can be any non-numeric character that does not appear in the item-id specification.

In this case, the delimiter cannot be a left parenthesis "(". A left parenthesis is used to designate a merge from a different file.

If either the "number of lines" (n) or the start position (s) is omitted, then a 1 is implied. If "item-id" is omitted, then MERGE occurs from within the current item.

Examples:

ME10//5	MERGE after the current line 10 lines from the current item starting at line 5.
ME//3	MERGE after the current line 1 line from the current item starting at line 3.
ME99/BOILER/1	MERGE after the current line 99 lines from the item BOILER starting at line 1 of BOILER.

Create a new item called MERGETEST in the SAMPLES file. MERGETEST will initially be an exact copy of ATEST in the SAMPLES file. The object is to MERGE the entire item ATEST into this new item.

Use item ATEST from file SAMPLES.

ATEST looks like this:

```
ID : ATEST
001 This is the first line
002 This is the second line.
003 This is the third line.
004 This is the fourth line.
005 This is the fifth line.
006 This is the sixth line.
```

Invoke EDIT for item MERGETEST in the file SAMPLES.

```
>ED SAMPLES MERGETEST <r>
NEW ITEM
TOP
.ME99/ATEST/1 <r>
.F <r>
TOP
.
```

MERGETEST is new.

MERGE 99 lines from ATEST starting at line 1 of ATEST into the beginning of MERGETEST. Since ATEST has only 6 lines, the entire item is merged.

Don't forget to (F)lip so the changes can be reviewed.

Review the item.

```
.L22 <r>
001 This is the first line
002 This is the second line.
003 This is the third line.
004 This is the fourth line.
005 This is the fifth line.
006 This is the sixth line.
EOI 006
```

List 22 lines.

End of item at line 6.

EDIT command prompt.

The item ATEST is now merged into the new item MERGETEST.

MERGE lines 3 and 4 to the bottom of MERGETEST.

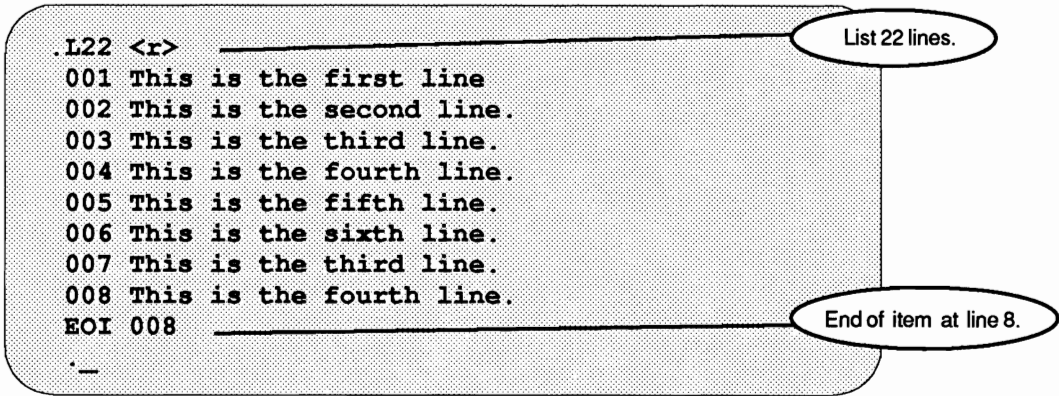
Since the line pointer is already at the bottom (line 006), continue.

```
EOI 006
.ME2//3 <r>
.F <r>
TOP
.
```

MERGE 2 lines starting at line 3 of this item. Notice that the two consecutive delimiters (//) specify a NULL item-id, indicating a MERGE from within the current item.

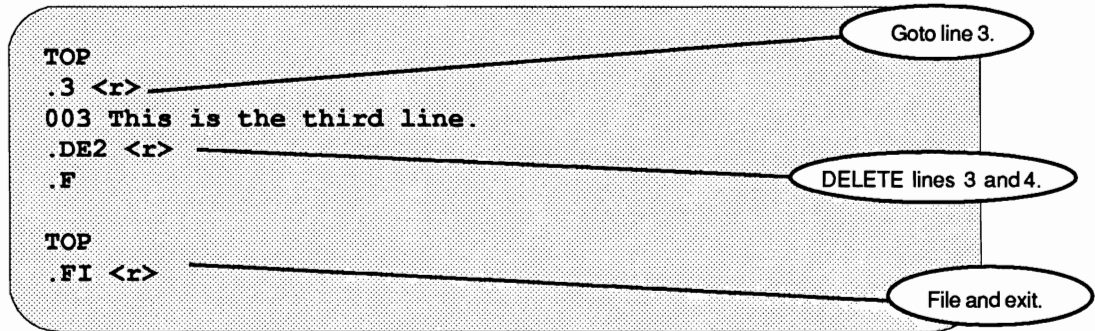
(F)lip the buffers.

Review the item.



The third and fourth line are merged to the end of item. Lines 3 and 4 are still intact at their original locations.

NOTE: MERGE is NOT a move. It basically means to copy and add. It does not delete the source lines. If you wish to MERGE and DELETE, you must perform the DELETE LINES (DE) command on the source lines following the invocation of MERGE.



Since MERGE performs an INSERT of lines, all the lines following MERGE are renumbered following the Flip.

9.3 Merging From A Different File

<u>Command</u>	<u>Description</u>
MEn(filename item-id)s	MERGE a number of lines (n) from the source item in the source file addressed by "(filename item-id)" starting at line (s) in the source item.

In this case, the delimiter MUST be a left parenthesis "(". This indicates the MERGE source item is in another file.

If either the "number of lines" (n) or the source start position (s) is omitted, then a 1 is implied.

If the item-id is omitted, then the MERGE occurs from an item in the source file using same id as the current item.

Examples:

ME5(MEMO THANKS)10	MERGE 5 lines from the item "THANKS" in the file MEMO starting at line 10 of "THANKS".
ME99(BACKUP)1	MERGE 99 lines starting at line 1 from an item in the file BACKUP. The source item has the same item-id as the current item.

Create a new item called MERGETEST2 in the SAMPLES file. MERGETEST2 is created by merging the entire item THANKS from the MEMO file.

```
>ED SAMPLES MERGETEST2 <r>
NEW ITEM
TOP
.ME99(MEMO THANKS)1
.F <r>
TOP
.
```

The item is new.

MERGE 99 lines starting at line 1 of the item THANKS in file MEMO to the TOP of the new item MERGETEST2 in the file SAMPLES.

Flip the buffers.

Review the item

```
.L22 <r>
001 Memo 01/01/87
002 To: Charlie
003 From: Me
004
005 Charlie,
006 This is to inform you that your gift was received
007 and greatly appreciated. A very merry to you too!
008
009 Thanks,
EOI 009
.
```

List 22 lines.

End of item at line 9.

EDIT command prompt.

9.4 Merging Errors

If the source item or file does not exist, the MERGE command will respond with the System Message,

```
NOT ON FILE
```

The source item cannot be found. Make sure the filename or item-id is not misspelled.

```
.ME99(MMO THANKS)1 <r>  
NOT ON FILE  
.
```

The filename MEMO is misspelled. Re-enter the MERGE command.

```
.ME99 (MEMO THANKS) 1 <r>  
.F <r>  
TOP  
.L22 <r>  
001 Memo 01/01/87  
002 To: Charlie  
003 From: Me  
004  
005 Charlie,  
006 This is to inform you that your gift was received  
007 and greatly appreciated. A very merry to you too!  
008  
009 Thanks,  
EOI 009  
.FI <r>
```

The sequence of MERGES, as in all changes, must be from the TOP of the item down. Changes must be made in ascending sequence of line numbers. If you enter out of sequence, the

SEQN?

error message is displayed. Flipping the buffer always resets the line pointer to the TOP of the item.

Exercise 12

1. Fill in the commands to MERGE 10 lines after line 10 of the current item, beginning with line 25 of the current item.

TOP

. _____
010 This is line 10 for a MERGE exercise
. _____
. _____

TOP

.

2. MERGE after line 5, lines 3 through 10 of item XYZ in the current file.

TOP

. _____
005 This is line 5.
. _____
. _____

TOP

.

3. Fill in the commands to MERGE to the BOTTOM of the current item, the entire item XYZ in the file MEMO. Assume that XYZ has 20 attributes.

TOP

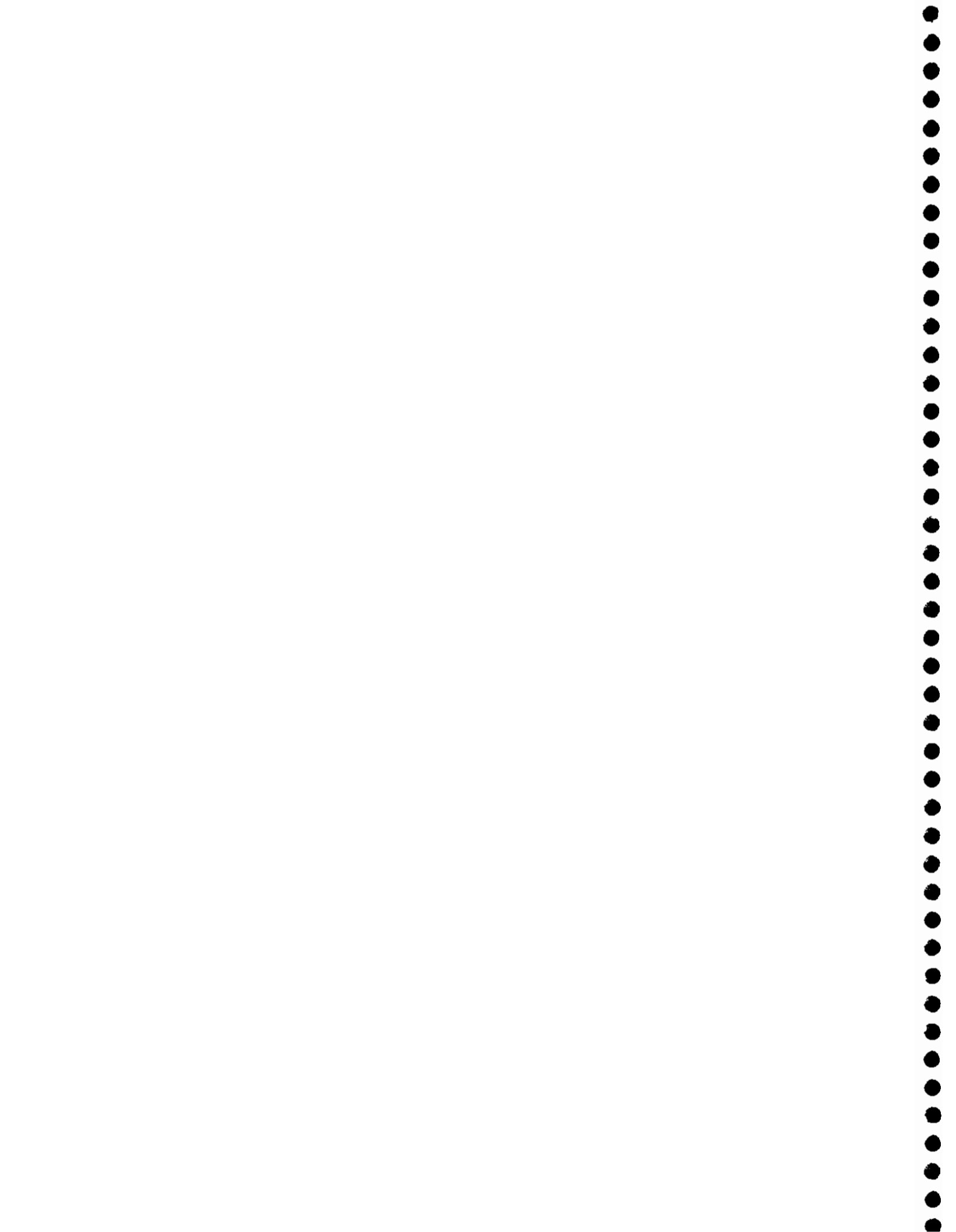
. _____
EOI 012
. _____
. _____

TOP

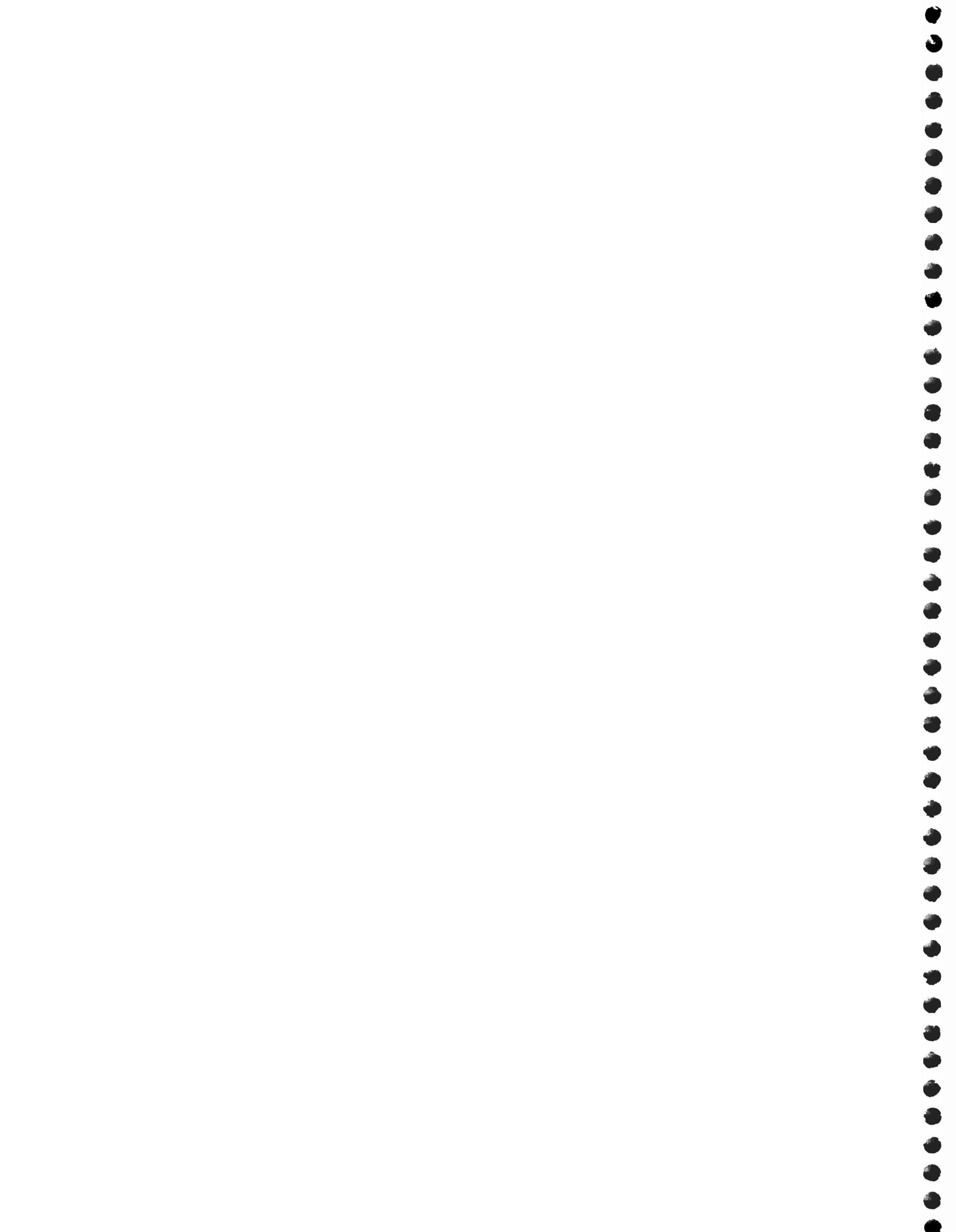
.

4. Match the command with its description.

_____ ME3//2	a.	MERGE 2 lines starting at line 2 of the item XYZ in the current file.
_____ ME//10	b.	MERGE 3 lines starting at line 2 of the current item.
_____ ME TESTID	c.	MERGE 10 lines starting at line 10 of the item TESTID in the SAMPLES file.
_____ ME9 TESTID 10	d.	MERGE line 10 of the current item.
_____ ME2/XYZ/2	e.	MERGE 2 lines starting at line 3 of an item in the SAMPLES file with the same id as the current item.
_____ ME2(SAMPLES)3	f.	MERGE 9 lines starting at line 10 of
_____ ME10(SAMPLES TESTID) 10		



NOTES



10. Cancelling Changes

It is often necessary to cancel a change or series of changes. Invoking the following CANCEL CHANGES (X) command allow you to delete the last change or all changes since the last (F)lip command. Changes in EDIT are any command which affect the data (INSERT, DELETE, REPLACE and/or MERGE).

Remember, if an (FI) or (FS) has been performed, the item has been saved and the (X) commands have no effect.

Always be sure to save a copy of the item under a new item-id before making changes in EDIT so that mistakes are recoverable.

<u>Command</u>	<u>Description</u>
X	CANCEL the effect of the last change (INSERT, DELETE, REPLACE, or MERGE).
XF	CANCEL the effect of all changes since the last (F)lip.
Related Command;	
F	Flip buffers. Reflect the last series of changes to the work item without writing to the file. This occurs automatically at the termination of INSERT in a new item. (See "SAVING YOUR WORK.")

10.1 The Last Change

Invoke EDIT on item THANKS in the file MEMO.

```
>ED MEMO THANKS <r>
TOP
.
```

Review it.

.L22 <r>
001 Memo 01/01/87
002 To: Charlie
003 From: Me
004
005 Charlie,
006 This is to inform you that your gift was received
007 and greatly appreciated. A very merry to you too!
008
009 Thanks,
EOI 009
.

List 22 lines.

End of item at line 9.

Replace the word "merry" with the word "happy" on line 7.

.7 <r>
007 and greatly appreciated. A very merry to you too!
.R/merry/hapi
007 and greatly appreciated. A very hapi to you too!
.

Go to line 7.

Replace "merry" with "hapi".

This is obviously a mistake. The word "happy" is misspelled as "hapi". CANCEL this change.

.X <r>
.F <r>
TOP
.

Cancel last change.

(F)lip the buffer to prove that the change is CANCELLED.

Back at the TOP.

Review the item now.

List 22 lines.

```
.L22 <r>
001 Memo 01/01/87
002 To: Charlie
003 From: Me
004
005 Charlie,
006 This is to inform you that your gift was received
007 and greatly appreciated. A very merry to you too!
008
009 Thanks,
EOI 009
.
```

End of item at line 9.

The change was cancelled.

10.2 All Changes Since The Last (F)lip

Invoke EDIT on item THANKS in the file MEMO.

```
>ED MEMO THANKS <r>
TOP
.
```

Review it.

List 22 lines.

```
.P <r>
001 Memo 01/01/87
002 To: Charlie
003 From: Me
004
005 Charlie,
006 This is to inform you that your gift was received
007 and greatly appreciated. A very merry to you too!
008
009 Thanks,
EOI 009
.
```

End of item at line 9.

Make 2 changes.

First, add the word "Hello" before "Charlie" on line 5.

```
.5 <r>  
005 Charlie,  
.R//Hello<space>/ <r>  
005 Hello Charlie,  
.
```

Second, DELETE (DE) line 9.

```
.9 <r>  
009 Thanks,  
.DE <r>
```

Now CANCEL the last 2 changes.

```
.XF <r>  
L000  
.<r>  
TOP  
.
```

Review the item.

```
.L22<r>  
001 Memo 01/01/87  
002 To: Charlie  
003 From: Me  
004  
005 Charlie,  
006 This is to inform you that your gift was received  
007 and greatly appreciated. A very merry to you too!  
008  
009 Thanks,  
EOI 009  
.FI <r>
```

List 22 lines.

End of item at line 9.

File and exit.

The last 2 changes were CANCELLED.

Exercise 13

1. What is the command to CANCEL all changes since the last FLIP? . _____
2. What is the command to CANCEL the last change? . _____
3. Circle the change commands which are affected by the CANCEL command in the following example.

```
>ED SAMPLES ATEST <return>
TOP
.G3 <return>
003 This is the third line.
.DE <return>
.5 <return>
005 This is the fifth line.
.R/is/was <return>
005 This was the fifth line.
.DE <return>
.X <return>
.F <return>

TOP
.
```

4. Circle the change commands which are unaffected by the CANCEL command in this example.

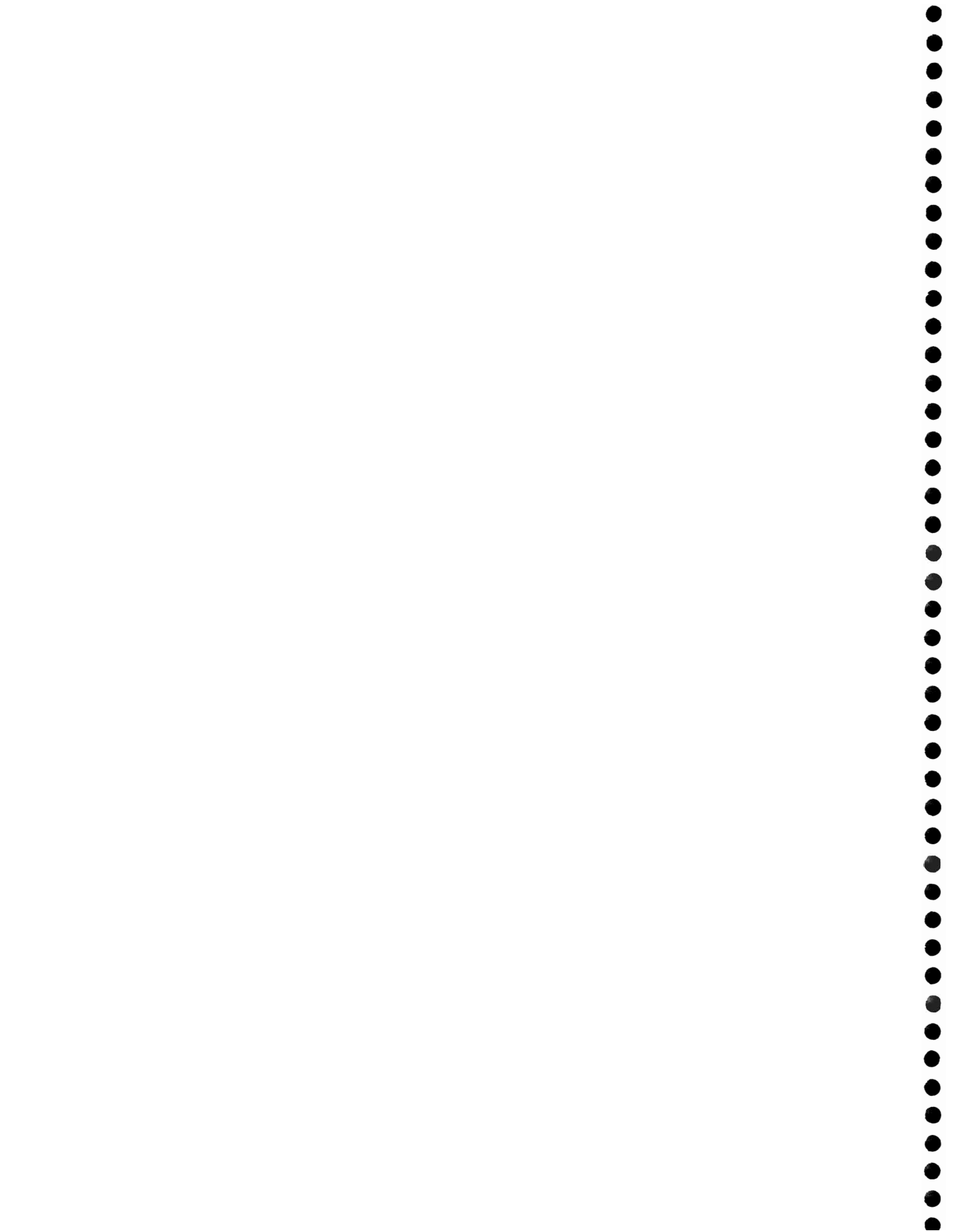
```
>ED SAMPLES ATEST <return>
TOP
.3 <return>
003 This is the third line.
DE <return>
. <return>
004 This is the fourth line. <
.R/This/That <return>
004 That is the fourth line.
.F <return>

TOP
.2
002 This is the second line.
DE <return>
ME100 (SAMPLES ANOTHER.TEST)1 <return>
.XF <return>
.F <return>

TOP
.
```



NOTES



11. Prestored Commands

Sometimes, when editing an item, a complex series of commands must be entered multiple times. Having to type these commands over and over again becomes tedious and time consuming.

PRESTORES allow a single command or complex series of commands to be temporarily stored in EDIT and executed using 2 or 3 keystrokes.

This section covers how to assign, display, and invoke PRESTORED COMMANDS>

11.1 Assigning Prestored Commands

There are 10 PRESTORE entries, numbered from 0 to 9. Each can hold a single or series of EDIT commands.

<u>Command</u>	<u>Description</u>
P number command	Assign a single EDIT command to a PRESTORE entry. "number" is the PRESTORE entry (0-9).
P number command<esc>command<esc>...	Assign a series of EDIT commands to PRESTORE entry referred to by "number". Each command defined in a PRESTORED series is separated by an ESCAPE character.

Attention Ultimate users: You cannot use the esc character as the command separator. Instead, you must use a <ctl><shift>\ as the command separator.

PRESTORES remain active while editing a single item or a series of items from an active item-list. Invoking EDIT from TCL resets PRESTORED entries to null.

The command "L22" (LIST 22 lines) is assigned to PRESTORE entry "0" by default upon initiation of EDIT.

The maximum number of characters in each PRESTORE entry is 100. If this is exceeded, the next entry is overwritten. For example, if PRESTORE entry 0 has 110 characters in it, PRESTORE entry 1 is not usable since its first 10 characters have been overwritten by entry 0.

Examples:

P1 R/This/That Assign the command that REPLACes the first occurrence of "This" with "That" to PRESTORE entry 1.

P2 RU999/10-10-86/01-01-87<esc>FI

Assign the following series of commands to PRESTORE 2. UNIVERSALLY REPLACE within 999 lines each occurrence of the date 10-10-86 with 01-01-87, FILE and EXIT.

PRESTORES can be used to re-invoke other PRESTORES. If an item-list is active, an FI or EX will EXIT the current item, EDIT the next item on the list and continue with the designated PRESTORE. The process will terminate once the last item on the list has been exited.

11.2 Displaying and Invoking Prestored Commands

<u>Command</u>	<u>Description</u>
PD	Display the currently active PRESTORED entries.
Pnumber	Invoke PRESTORE entry referenced by (number).

EDIT the item THANKS in the MEMO file and take a look at how a PRESTORE might work.

Invoke EDIT.

>ED MEMO THANKS <r>

TOP

.PD <r>

P0 L22

.P <r>

001 Memo 01/01/87

002 To: Charlie

003 From: Me

004

005 Charlie,

006 This is to inform you that your gift was received

007 and greatly appreciated. A very merry to you too!

008

009 Thanks,

EOI 009

.P1 6[R/inform/let you know/[7[R/very/very very/[F[P<return>

.PD<return>

P0 L22

P1 6[R/inform/let you know/[7[R/very/very very/[F[P

.P1<return>

006 This is to inform you that your gift was received

006 This is to let you know that your gift was received

007 and greatly appreciated. A very merry to you too!

007 and greatly appreciated. A very very merry to you too!

TOP

001 Memo 01/01/87

002 To: Charlie

003 From: Me

004

005 Charlie,

006 This is to let you know that your gift was received

007 and greatly appreciated. A very very merry to you too!

008

009 Thanks,

EOI 009

Display what is currently in the PRESTORES.

PRESTORE 0 has the default, List 22 lines.

Invoke PRESTORE 0. Notice the 0 is implied. This lists 22 lines.

Assign a series of commands to PRESTORE 1.

The left bracket ([) is what the system echoes when the <escape> key is pressed.

Invoke PRESTORE 1.

This says, Goto line 6, REPLACE "inform" with "let you know", Goto line 7, REPLACE "very" with "very very", (F)lip the buffers, and invoke PRESTORE 0. Prestore 0 lists 22 lines.

End of item at line 9.

The invoked PRESTORE performs all the designated commands.

11.3 Using Item-Lists

PRESTORES can be invoked to take action on every item in an Item-List. This process is initiated by having the PRESTORE invoke a command which exits the present item and then re-invokes itself or another PRESTORE.

Example:

Write a PRESTORE to DELETE line 2 in all items on the Item-List.

Invoke EDIT for items A, B, C, D, E, F and G on the SAMPLES file.

```
>ED SAMPLES A B C D E F G <r>
```

```
A
```

```
TOP
```

```
._
```

```
.P1 G2<esc>DE<esc>FI<esc>P1 <r>
```

```
.PD <r>
```

```
P0 L22
```

```
P1 G2[DE[FI[P1
```

```
.P1 <r>
```

```
002 This is line 2
```

```
'A' FILED
```

```
B
```

```
TOP
```

```
002 This is line 2
```

```
'B' FILED
```

```
C
```

```
TOP
```

```
002 This is line 2
```

```
'C' FILED
```

```
D
```

```
TOP
```

```
002 This is line 2
```

```
'D' FILED
```

```
E
```

```
TOP
```

```
002 This is line 2
```

```
'E' FILED
```

```
F
```

```
TOP
```

```
002 This is line 2
```

```
'F' FILED
```

```
G
```

```
TOP
```

```
002 This is line 2
```

```
'G' FILED
```

```
>
```

Assign the PRESTORE command.

PRESTORE Entry 1 says: Go to line 2, DELETE it, FILE and EXIT, and re-invoke PRESTORE 1. This continues until all the items in the item-list are altered.

Display the PRESTORE entries.

Invoke PRESTORE number 1.

Back at TCL.

Line 2 in every item in the Item-List has been deleted!

Exercise 14

1. What is the default value of PRESTORE 0? _____
2. What is the command to display all PRESTORE Entries? _____
3. Assign the command "R/X/Y" (REPLACE X with Y) as PRESTORE 9. _____
4. Assign the command "ME3//2" (MERGE 3 lines starting at line 2) to PRESTORE 1.
• _____
5. Fill in the system response for this command if the previous assignments have been made.

 .PD <return>

6. Assign the following sequence of commands to PRESTORE entry 3.
 - a. List 22 lines
 - b. Delete line 10
 - c. On line 5, REPLACE the first X with a Y
 - d. FILE and EXIT
 - e. Re-invoke PRESTORE 3.• _____
7. EDIT and FILE DELETE items A, B, C, D, E, and F on the SAMPLES file with a single PRESTORE.

 Invoke EDIT

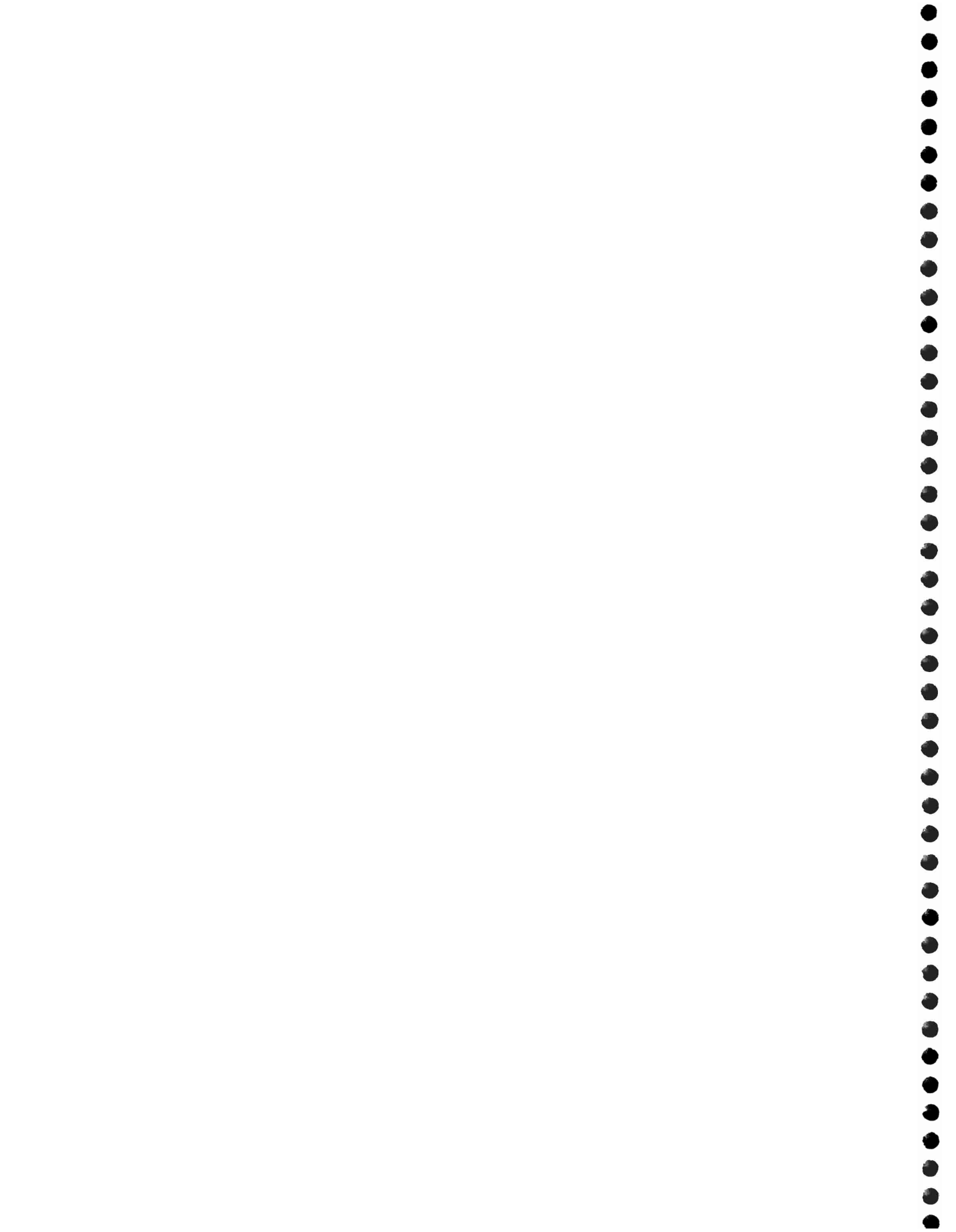
 > _____
 A
 TOP
 .

 Assign the commands to perform the FILE DELETES to PRESTORE 5.

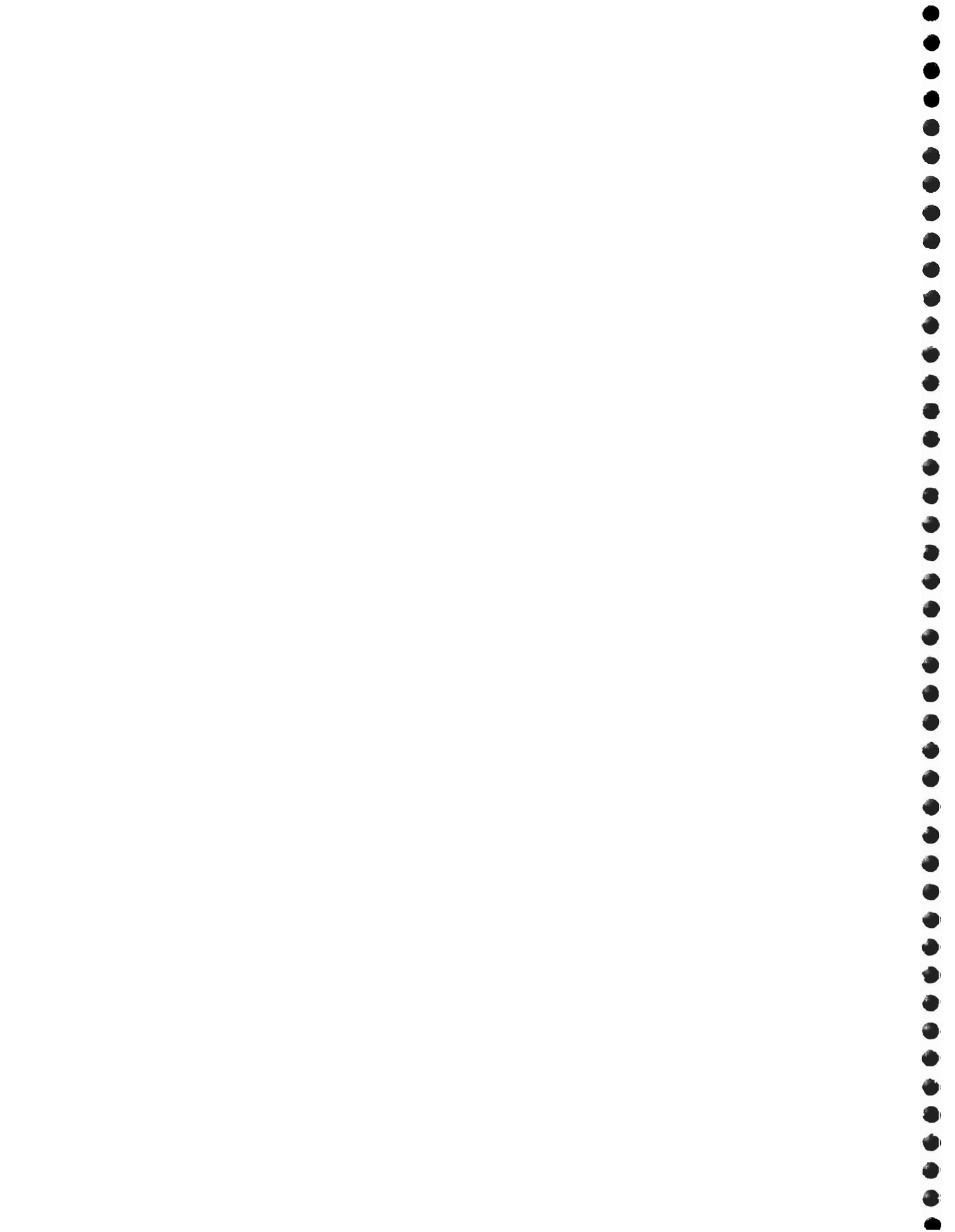
 • _____

 Invoke PRESTORE 5.

 • _____



NOTES



12. Other Commands

These commands are provided here for your information only. You don't need to do any of the examples.

<u>Command</u>	<u>Description</u>
AS	Toggles Assembler Format on and off. Assembler Format arranges the display of an Assembler Source item so that it is more readable.
M	Toggles Macro Expansion on and off. Many Assembler mnemonics are "macros," a set of "lower-level" assembler statements to perform a complex function.
S	Toggles Suppress on and off. Suppress the display of line numbers, or Suppress the display of imbedded object code in an Assembler Source item.
Zbegcol-endcol	Zone Display. Display data or text which exists between the begcol and endcol.

12.1 Assembler Format

EDIT is invoked for item GETINPUT in the file ASM

```
>ED ASM GETINPUT <r>
```

```
TOP
```

```
.9 <r>
```

```
009 EP !GETINPUT\0005 000002C0] JUS @01
```

```
.L7 <r>
```

```
010 EP !PATCH\0009 000002C0] JUS @01
```

```
011 EP !UNPATCH\000D 000002C0] JUS @01
```

```
012 *
```

```
013 !GETINPUT EQU *\0011 000000C0
```

```
014 ZERO D1\0015 000008C2] XOR AX,AX] STA @B1] STA @B1(2)]@W1
```

```
015 ZERO D0\0019 000008C2] XOR AX,AX] STA @B1] STA @B1(2)]@W1
```

```
016 !LOOP1 EQU *\001D 000000C0
```

```
.
```

```
.AS <r>
```

Go to line 9.

List 7 lines.

Turn on Assembler formatting.

Goto line 9 and review same lines.

```
.9 <r>
009 0009 000002C0          EP    !GETINPUT
.L7 <r>
010 0009 000002C0          EP    !PATCH
011 000D 000002C0          EP    !UNPATCH
012 *
013 0011 000000C0    !GETINPUT EQU    *
014 0015 000008C2          ZERO    D1
015 0019 000008C2          ZERO    D0
016 001D 000000C0    !LOOP1  EQU    *
.
```

Goto line 9.

List 7 lines.

The Assembler listing is formatted in a easier to read fashion, with the object code first.

12.2 Suppress Line Numbers and Object Code

Now, Suppress the line numbers and object code.

```
.9 <r>
009 0009 000002C0          EP    !GETINPUT
.S <r>
.L7 <r>
010          EP    !PATCH
011          EP    !UNPATCH
012 *
013 !GETINPUT EQU    *
014          ZERO    D1
015          ZERO    D0
016 !LOOP1  EQU    *
.
```

Goto 9.

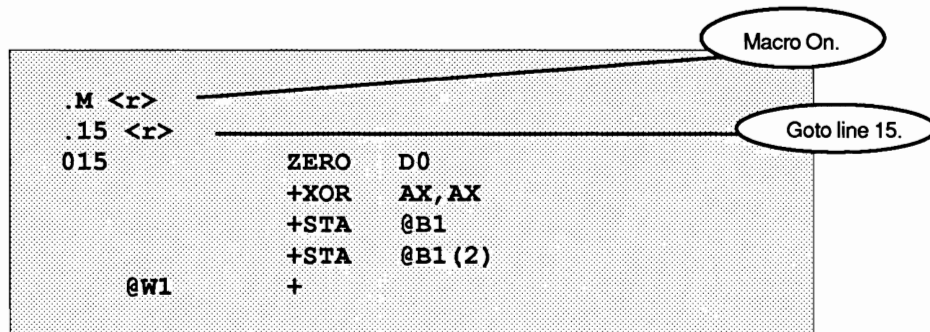
Suppress On.

List 7 lines.

Both the line numbers and object code displays are suppressed.

12.3 Macro Expansion

Look at the line 15 with Macro Expansion on.



Line 15 "ZERO D0" is made up of the commands preceded by plus signs (+).

12.4 Zone Display

Invoke EDIT on the item ATEST in the SAMPLES file.

```
>ED SAMPLES ATEST <r>
TOP
.
```

Review it.

```
.L22 <r>
001 This is the first line.
002 This is the second line.
003 This is the third line.
004 This is the fourth line.
005 This is the fifth line.
006 This is the last line.
EOI 006
.
```

List 22 lines.

Go to the TOP and set up a Zone Display of columns 13 through 17.

```
.T <r>
```

```
TOP
```

```
.Z13-17 <r>
```

Set the display zone.

Review the item.

```
.L22<r>
```

```
001 first
```

```
002 second
```

```
003 third
```

```
004 fourth
```

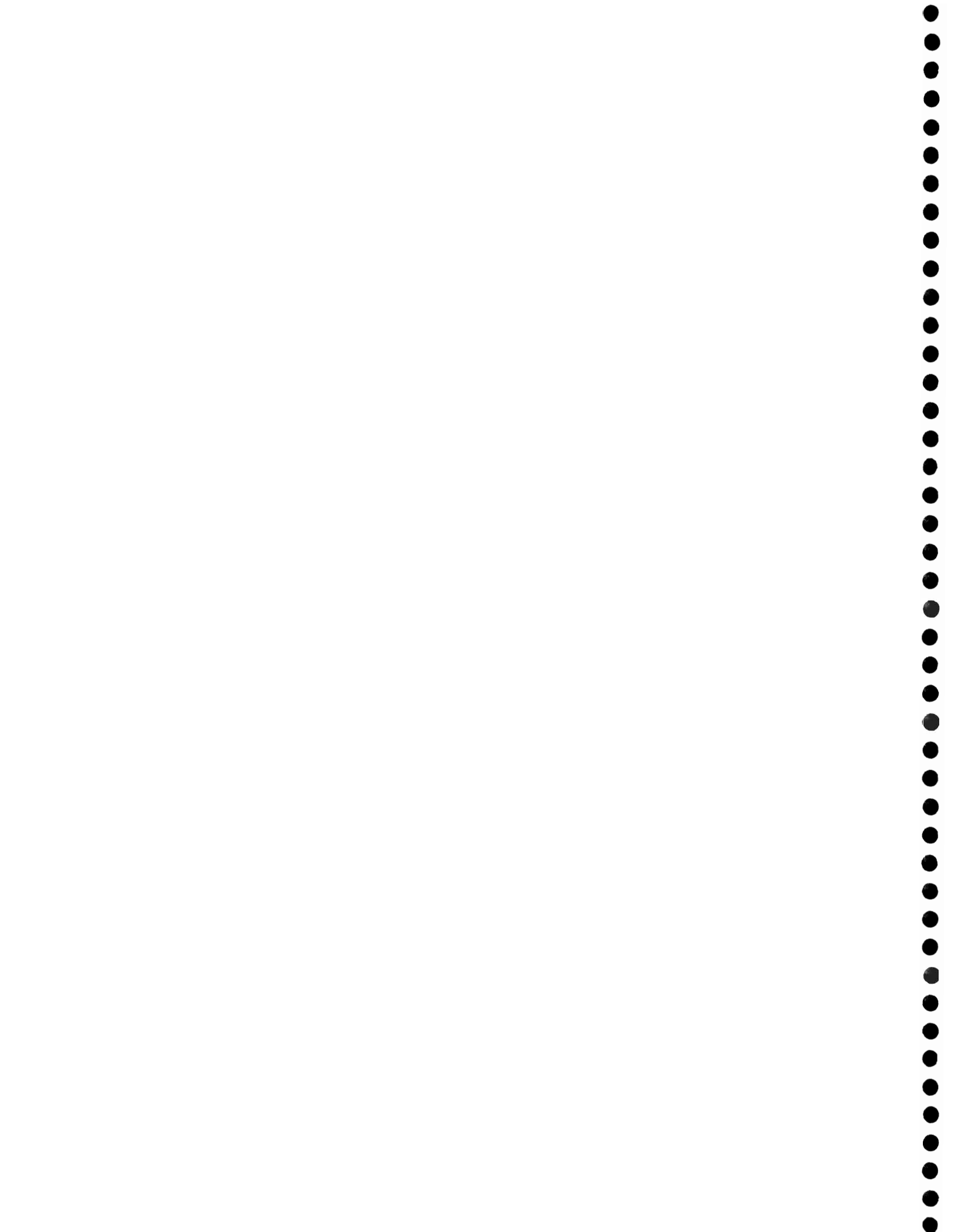
```
005 fifth
```

```
006 sixth
```

```
EOI 006
```

```
.
```

NOTES



Glossary

absolute positioning	A command to take you to a specific point in the item.
account	A directory of information and services within the system dictionary.
append	Inserting text to the end of an existing line in an item.
ASCII	Stands for American Standard Code for Information Interchange.
attribute	A field within an item delimited by an attribute mark. (ASCII character 254 represented by an up-arrow "^").
byte	A collection of bits (8 in this case) which represent a character under some arbitrary standard (ASCII in this case).
bit	A binary unit. The smallest piece of information in a computer. A bit has only 2 states, on or off, represented by a 1 or a 0. In this case, 8 bits make up 1 byte.
columnar zone	A segment of the line in an item bounded by beginning and ending columnar positions.
cursor	A symbol on terminal displays to indicate a point on the screen where operator input or system output may occur.
data	A collection of characters which represent information to be displayed or processed.
DATA file	The "lower" level of a file, containing the data and/or text which make up the file's information.
delimiter	Any character which separates the elements of a data record or a command.
DICT file	The "upper" level of a file, containing DATA definition items used by ACCESS for retrieving and formatting data.
editor	A program for revising the contents of an item.
file	A set of distinct elements of information grouped according to purpose, located within an account
item	The individual elements in a file, consisting of multiple pieces of information.
item-id	System-assigned identification used to uniquely identify items and their physical location.
line editor	A type of editor which allows addition and revision on a line-per-line basis. Originally devised as an editor to be used on outdated teletype devices.
line pointer	An index used internally by EDIT to keep track of the current attribute of the item being worked on.

master dictionary	A file containing those elements defining valid files and commands in the account.
multi-value	A field within an attribute delimited by a value mark, (ASCII character 253 represented by a right bracket "]").
prefix	Inserting text at the beginning of a line in an item.
relative positioning	A command allowing you to address other lines in the item by skipping forward and backward from the current line.
sub-value	A field within a value delimited by a sub-value mark. (ASCII character 252 represented by a back-slash "\").
TCL	Terminal Control Language. The process which interprets and executes all command sentences. You know TCL is active when the prompt character (>) is displayed.
text	A string of characters which make up a descriptive statement. May be a line of a memo item, or a descriptive attribute of a data item.
truncate	To cut the end off a line.
verb	A word of action. The first word of a command sentence initiated at TCL.

Exercise Answers

Exercise 1

1. Account
2. System Dictionary
3. Master Dictionary
4. File Dictionary
5. DATA portion
6. Item-id
7. Attributes
8. Attribute Mark
9. DATA item, Text item, PROC item
10. infinite
11. 32,000

Exercise 2

1. logon
2. Terminal Control Language
3. verbs
4. LISTFILES
5. LIST filename
6. SORT filename
7. EDIT INVENTORY A20
8. "NEW ITEM"
TOP
9. The editor is awaiting for a command
10. EX
11. at TCL
12. A1 A2 A3
EX
EX
EX
13. EXK

Exercise 3

1. ED CUSTOMERS 300
TOP
2. line pointer
3. 003
4. B
2
4
T
EX
5. B
U4
N3
N1 or <return>
055
6. T
L4
4
7. S?

Exercise 3, continued

8. infinite
9. CUSTOMERS 100 L 004
10. L22

Exercise 4

1. C, I, D, A, J, B, L, G, M, K, H, F, E
2. ED SAMPLES EXERCISE4
I
This is a test of
how much you have
learned in the ADD LINES
Section of the EDITOR
Workbook.
<return>
3. L22 <return>
4. EDIT SAMPLES ATEST
L22
2
I
put any text here.
put any text here also.
<return>
F
5. FI
6. Inserting out of sequence, not from top to bottom.

EXERCISE 5

1. an empty line
2. return key
one
3. a character used to hold a line as a null line.
4. EDIT MEMO WELCOME
4
I
X
X
<return>
F
R10/X// <return>
5. EDIT MEMO WELCOME
4
^^
F

Exercise 6

1. TB
2. Tab
Control I
3. C
4. EDIT SAMPLES TABTEST
5
C
TB 11 23
I
JASON <TAB> 250 <TAB> 06-01-87 <rreturn>
<return>
F

Exercise Answers - page 2

Exercise 7

1. F, E, H, B, A, I, G, D, C
2. ED SAMPLES ATEST
L22 or L3
FI BACK.ATEST
3. EX
FI
FS BACK.3
EXK

Exercise 8

1. FD
2. RECOVER-FD
3. F
T
F
F
4. FD
RECOVER-FD ITEM.DELETE.TEST
5. FD
FDK

Exercise 9

1. 20
DE
F
2. 15
DE6
F
3. DE50
F
4. C
5. EDIT SAMPLES DTEST
DE5/test
6. 1, 3, 5
DE5/s/1-4

Exercise 10

1. L/HELLO
A
2. L120/123
120
3. L20/ABC
4. L120/X/1
5. L120/XYZ/1-10
6. ^
L200/^
7. H, G, F, I, B, E, D, A, C

Exercise 11

1. EDIT SAMPLES ATEST
2
R
This is replaced line 2
F
2. 3
R3
These are the replaced lines
These are the replaced lines
These are the replaced lines
F
3. 6
R/This is/That was
F
4. R6/This/That
RU6/This/That
5. RU6/A/*/10
6. 1
R//START /
F
7. G, F, J, H, I, D, C, A, E, B

Exercise 12

1. 10
ME10//25
F
2. 5
ME8/XYZ/3
F
3. B
ME20/XYZ/1
F
4. B, D, G, F, A, E, C

Exercise 13

1. XF
2. X
3. .DE <return>
4. DE <return>
.R/This/That <return>

Exercise 14

1. L22
2. PD
3. P9 R/X/4
4. P1 ME3//2
5. PO L22
P9 R/X/4
P1 ME3//2
6. P3 L22 [10[DE[F]5[R/X/Y]FI[P3
7. ED SAMPLES A B C D E F
P5 FD[P5
P5

TO ORDER ADDITIONAL COPIES OF *THE EDITOR*

Fill out this order form today and send it along with your payment to the most convenient IDBMA location.

PRICE:

\$14.95 (USA, Mexico and Canada) *
A\$25.50 (Australia) **
£10.00 (United Kingdom) **

* USA, Canada and Mexico residents add \$2.50
for shipping and handling. CA residents add 6% sales tax.

** UK/Australia prices include shipping and handling.

IDBMA, Inc.
World HQ
10675 Treena Street
Suite 103
San Diego, CA 92131
Tel: 619/578-3152
Fax: 619/271-1032

IDBMA, Pty. Ltd
Australia/Pacific
133 Alexander Street
Crows Nest, N.S.W. 2065
Tel: 02/439-5488
Fax: 02/439-2738

IDBMA S.A.
France
90 rue de la Victoire
75009 Paris
Tel: 1/4281-1730
Fax: 1/4526-9533

IDBMA Ltd.
United Kingdom
Holly Road
Hampton Hill, Middlesex
TW12 1PZ
Tel: 01/783-0055
Fax: 01/783-1678

THE EDITOR WORKBOOK ORDER FORM

NAME _____

COMPANY NAME _____

ADDRESS _____

CITY/STATE/ZIP _____

PHONE _____ FAX _____

Quantity _____

Subtotal _____

Shipping _____

Tax _____

Total _____

☐ Payment Enclosed ☐ Mastercard ☐ VISA ☐ AMEX

Card Issued To: _____

Card Number: _____

Cardholder Signature: _____





