THE IDIOT'S GUIDE TO TSS

1. Find a free teletype.

The four teletypes in room 225 Campbell Hall (phone 2-5008) and the four teletypes in room 125 T-7 (phone 2-0105) have been reserved for the use of CS 120A students every week day from 4:45 to 5:45. If anyone else is using them during this hour you may politely ask them to leave as soon as possible.

2. Setting the switches.

Now that you've found a free teletype you'll have to set a few switches properly. There are two kinds of teletypes around, 33's and 35's. A 33 is smaller and is yellow and brown, while a 35 is gray. The on-off switch on each is located near your right hand; make it point to LINE (LOCAL makes your teletype behave like a typewriter that doesn't transmit anything). The 35's usually have a button on your left hand which controls the paper tape gizmo. Make it point at K and forget about paper tape. If your teletype has a little toggle switch which points to either A or B, make it point to B - TSS always runs on the B machine.

3. Typing commands.

You should think of your communication with TSS as a series of commands, typed by you, which cause things to be done and evoke various responses, typed on your teletype by TSS. If you have hang-ups about giving commands think of them as requests. Each command is typed as a bunch of non-blank characters followed by a carriage return (the button marked RETURN) on the right side of the keyboard. TSS does not really start to act on your command until you type the carriage return, so you can correct mistakes as you go along any time before you press the return button. If you mistype a character you can correct it (immediately after it was typed) by typing CONTROL-Q (i.e. by holding down the button marked CTRL while typing Q). A left-arrow will be typed on your teletype; then you may type the character you intended in the first place. Typing two CONTROL-Q's will erase the last two characters, etc. If the line you're typing is so botched up that you want to start over, type CONTROL-Y. The teletype will print an up-arrow and perform a carriage return. There aremany other fancy tricks one can play with the CTRL button, but we won't discuss them here. (See the document on the line editor.)

4. Establishing contact.

The maneuver about to be described calls for dexterity (sinistro-dexterity, actually). With your left hand hold down the buttons marked CTRL and SHIFT while you strike a P with your right. The teletype should spring to life and print

```
BEAD VERSION 4.0
ENTER USER NAME
```

If it prints, instead,

```
PANIC ACKNOWLEDGED, BEAD HERE
```

that means that your teletype was being used by another student before you appeared. If neither of the above messages appears, there are two likely possibilities:

1) TSS is down. (If there is someone beside you happily typing away at TSS you know this can't be the case.)
2) Your teletype is out of order. Go back and check to see that all the buttons are set correctly. Try the foregoing procedure on another teletype, if oneis free, and if it works you know the first one is sick.

In either of the above cases, bring the problem to the attention of the T.A. (He probably won't be able to do anything, but you'll feel better having someone to talk to.)

If, however, TSS has written you a message indicating the presence of the BEAD you're in good shape and should type (whether TSS has requested it or not)

```
USER,idiot
```

where "idiot" is a 2-6 character string representing your name. (You should insure, somehow, that your name is distinct from that of any other student using TSS at the same time.) Don't forget the carriage return at the end. Also don't type any spaces.

Notice that "idiot" was written in small letters. That means you're not supposed to type IDIOT but rather your own name. This convention will be followed throughout: wherever upper case letters appear you should type them exactly as they appear on the page; wherever lower case letters appear they give only a general indication of the sort of information to be supplied -- you must provide the specific information (in this case your own name) yourself.

After you've typed

```
USER,idiot
```

TSS might type back

```
dummy WAS ONE
```

which means that "dummy" was using this teletype and

(a) is off going to the bathroom or something and will be back shortly, or

(b) failed to log out (see the logout procedure at the end of this guide) and will not be back.

If dummy comes back within five minutes it is polite to let him use the teletype, but let's get on with it anyway -- assuming case (b). In any event you will get a response saying

```
REMEMBER TO LOGOUT
```

and you should when you're through for the day.

5. Meet the BEAD.

You are now talking to the BEAD which performs a sort of ushering function taking you to the various sub-systems in which you may be interested. The reasons for its name are obscure; Cerebrus or Madam might have been better given its function and disposition. In any event you will be talking to the BEAD a great deal, asking it to do all sorts of different tasks. In order to do anything, however, you will need memory space (ECS space, in fact) to do it in. So the first thing you should ask the BEAD to do is to get you some space. This is done by typing

```
SPACE,n
```

where n is a number, like 10000. (The amount of space you should ask for depends on what you need to do.) The response to this request will be either OK , meaning that your request has been granted, or TOO MUCH meaning that it hasn't. Presumably you can't go on without space so you can either (a) log out and go home, (b) wait for a few minutes and type

```
SPACE,n
```

again, (c) reduce the size of n (if you think you can manage with less), or (d) ask one of your fellow time-sharers to give up some space. Unlike time, space cannot be shared, apparently.

In order to release space you type

```
    TRIM
```

This reduces the amount of space you have to what you are using at the moment (i.e. takingup with files). Whenever you are about to ask the BEAD to call a sub-system you should always start by typing TRIM (to release any extra space you may be holding) and then SPACE,n where n is only the amount needed by the particular subsystem you are about to call. An appropriate value for n is provided with the description of each sub-system.

6. Creating a file.

At last you're ready to do something! How about writing a Snobol program? The easiest way to do this is to create a file with the text editor - the file will consist of a bunch of lines which are the statements of your Snobol program. To get the editor type

```
    TRIM
    SPACE,5000
    C,EDITOR,S,fname,uname
```

where fname is the name of the file (2-6 characters) and uname is the user name attached to it. (If uname is omitted, the user name you signed on with is assumed.) If such a file already exists in ECS, the editor will operate on it; if no such file exists the editor will create it for you. In either case the editor will type EDIT just to tell you it's there. You can now start creating a new file by typing I followed directly by a carriage return. This will put the editor in insert mode and allow you to insert as many lines as you like. (In this case we are "inserting" them at the top of an empty file.) You can now type the statements of your Snobol program, starting labels in the first print position and being careful not to use more than 72 columns. (It is not very obvious on the teletype just where column 72 is.) Continuation lines may be employed as usual. Each line must be terminated with a carriage return.

When you've entered your entire program (remembering that the last statement says END starting int he first position) your file is complete. In order to explain that to the editor type a carriage return at the beginning of the next line. This signals the editor that you're leaving insert mode and you now want it to pay attention to (act upon) the command you will give it next. If you want the editor to make a printed listing of the file you just created, type

```
T;P$
```

followed, as usual, by a carriage return. The T will position the editor at the top of the file, and the P$ will instruct it to print all the lines until it comes to the end of the file (the last line of the file is symbolized by $). You can then see if any corrections that you made (using CONTROL-Q or CONTROL-Y) were indeed made correctly. If you're satisfied with the looks of your file type F (for finished and file) and the editor will send you back to the BEAD, leaving your newly created file in ECS where further things may be done to it. (More good news about the editor may be found in Section 12.)

7. Writing TSS Snobol.

TSS Snobol is exactly the same as the CAL Snobol you have been using, except for the addition of the two procedures IN() and OUT() which allow information to come in from and go out onto theteletype. Writing

```
OUT('HELP')
```

in your program will cause HELP to be printed on the teletype whenever the above statement is executed. Writing

```
CARD = IN()
```

will cause an up-arrow to be printed on the next line of the teletype paper and everything will be suspended while the program waits for a line of input to be given it from the teletype. Such a line must, as usual, be terminated by a carriage return. IN() never has anything but a null argument; the argument of OUT() may be any expression which yields a string when evaluated.

The use of the name OUTPUT as in

```
OUTPUT = 'HELP'
```

will cause characters to be sent to the file named OUTPUT as usual. In addition the predefined procedure OUTPUT() may be used as in

```
OUTPUT('PRINT','FILE1','0')
```

to cause the execution of

```
PRINT = 'HELP'
```

to send the characters OHELP to the file named FILE1. (The zero is to be used for carriage control -- double space -- when the contents of FILE1 are printed.

There is no way of controlling the carriage of the teletype except by executing a statement of the form

```
OUT()
```

to produce a blank line.)

If data is to be read into the program from some file resident in ECS rather than from the teletype, an INPUT() procedure of the form

```
INPUT('READ','DATA','80')
```

should be used, rather than the name INPUT . If your data has been stored on a file named DATA (which has perhaps been generated by the editor just as your Shobol program file was generated and which is now in ECS under your user name), then execution of

```
NEXT = READ
```

will give NEXT the value of the next line in the DATA file.

8. Using the disk.

It's a good idea to save files on the disk whenever you get through playing with the editor. In fact, if you're editing a very long file its a good idea to leave the editor periodically -- say every 100 lines or so -- to preserve your work on the disk. Then if something happens to the system your work will not be lost in the crash but will be safely stashed away where you may retrieve it at another time. In order to ask the BEAD to call the DISK for you, type

```
TRIM
SPACE,6000
C,DISK,S
```

and the disk should respond by saking DISK HERE.

If this is the first time you're saving the file you must create a disk entry for it. This is done by typing

```
CREATE,fname1,uname1,fname2,uname2
```

where fname1,uname1 are the file and user names of the ECS file you wish to preserve, and fname2,uname2 are the file and user names to be given the file when it is stored on the disk. If these sets of names are to be the same (which is usually the case) then only fname1,uname1 need be given. (Both uname's are assumed to be the same as the name you logged in with if they are omitted.)

The disk will respond by typing CREATED if it succeeds in its task, or ECS FILE EMPTY if it fails. This probably means that you inadvertantly typed fname1 wrong, giving the name of a file that has no lines in it. You can then do the CREATE again properly.

Having created a disk entry you can now ask the disk to put your file safely away by typing

```
PUT,fname1,uname1,fname2,uname2
```

The disk should respond by typing MOVED . If instead it responds FILE NOT ON DISK this means that this is a new file and you have forgotten to do a CREATE for it, or have just made another typing error.

When the DISK puts a file away for safekeeping it removes it from ECS. If you wish to do more work with it now (execute its statements, for example), you must get it back again. This is done by typing

```
GET,fname1,uname1,fname2,uname2
```

where, as before, fname1,uname1 refers to the name to be given the file when it is brought into ECS, and fname2,uname2 refers to the name by which the file is identified on the disk. Note that it is sometimes convenient to bring a file into ECS by a name different from that by which it is identified on the disk; you may wish, for example, to copy most of a large file, then add afew lines with the editor, and then store the modified file on the disk by a new name, leaving the old version stored also.

The disk should respond OBTAINED . If it says instead FILE NOT ON DISK then eitheryou didn't PUT it properly or you didn't GET it properly - try again.

Note that when you PUT a file on the disk it destroys any old file of that same fname2,uname2 that may be residing there. You will often want to "write over" old files - for example as you create successively more accurate versions of your program - but be sure that's what you intend when you do a PUT . You may then GET this file from the disk at any later time (or day) - and in fact you will often start a session at the teletype not by creating a new file with the editor but by getting an old file that has been saved for you on the disk.

If the disk doesn't understand one of your commands it will print out SAY AGAIN and you should indeed say it again. If the disk fails to respond to any of your commands it may be 'locked up'. As disastrous things may happen if you tell it the wrong thing at this time, stop trying to tell it anything and talk to the T.A. instead. (He may not be able to help, but at least you won't responsible for anything bad that happens next. See the warning in Section 15)

9. Keeping the disk clean.

You are encouraged to clean up the disk periodically by destroying any old files you are no longer going to use. While you are talking to the disk you can ask it to print a list of all files stored under your user name by typing

```
LIST,uname
```

It will produce a list of file names followed by the message LIST COMPLETE . If, when you read this list, you find the names of now useless files type

```
FLAGON,DELETE
```

to which the disk will respond ON telling you that it has turned on a flag which allows you to perform deletion. You can then type

```
DELETE,fname,uname
```

for any file that you choose. (Please make an effort to restrict yourself to deleting only those files that belong to you.) The disk will re spond DELETED for each file that it discards; if instead it types FILE NOT ON DISK it probably just means another damn typing error. Try it again.

When you're all through talking to the disk about all of your problems, type FIN and it will send you back to the BEAD.

10. Executing a Snobol program.

Now we've created a file which consists of a Snobol program, stored it on the disk, and brought it back into ECS. How about executing it? In order to call the Snobol compiler, we first have to ask the BEAD to call SCOPE . This is done by typing

```
TRIM
SPACE,55000
C,SCOPE,S
```

SCOPE should respond with

```
TSS SCOPE VER. 2.06
ENTER TIME AND DATE
```

You can happily ignore this last request and type instead

```
TEXT,OUTPUT
```

which will insure the fact that the file named OUTPUT , which will soon receive the listingof your program and any output you may send to the file named OUTPUT , will be in a format which can be printed. You should get a message back saying GENERATED and then proceed to call Snobol by typing

```
X,SNOBOL,I=fname
```

where fname is the name of the file which consists of your Snobol program. Note that this command says I=fname rather than I=fname,uname since SCOPE's file handling system uses only one parameter (fname) rather than two. You should thus be sure that the file you are to execute has a uname which is the same as the one you logged on with, because that uname is assumed. (If you need to work with a file under SCOPE which has someone else's user name, you may type

```
GET,fname,uname
```

to SCOPE (not to the disk). This will put the file into SCOPE's file directory under the filename only so you can refer to it with this single parameter.)

Snobol will then proceed to compile your program, if possible. If there are no compilation errors in your program, Snobol will print SUCCESSFUL COMPILATION and proceed to go into execution. If the program terminates successfully (i.e. if there are no execution errors) Snobol will type END and return control to SCOPE.

The listing of the program and (probably) the results from its execution will all be found on the file named OUTPUT so you will have to invoke the editor in order to see what has happened. To do this you should type DONE to tell SCOPE that you no longer need its services and it will return you to the BEAD. You should then type TRIM to release all that space that SCOPE needed, and then type

```
SPACE,5000
C,EDITOR,S,OUTPUT
```

You can then do a P$ to print the entire file,

More likely, you will have encountered an error of some sort. If your program did not compile successfully, SCOPE will type out ABORT . If your program did compile successfully but failed to executed properly, then Snobol will print a message of the form

```
ERROR TERMINATION IN RULE m AT LEVEL n
```

In either case you should type DONE to SCOPE which will return you to the BEAD. You can then call in the editor to look through your output file and see what the trouble is.

11. <u>Leaving sorcerer's apprentice mode</u>.

If your Snobol program goes into an infinite loop (enters sorcerer's apprentice mode) you should always panic - not psychologically but physiologically. That is, should should hit CONTROL-SHIFT-P which should wake up the BEAD and cause it to print

```
PANIC ACKNOWLEDGED - BEAD HERE
```

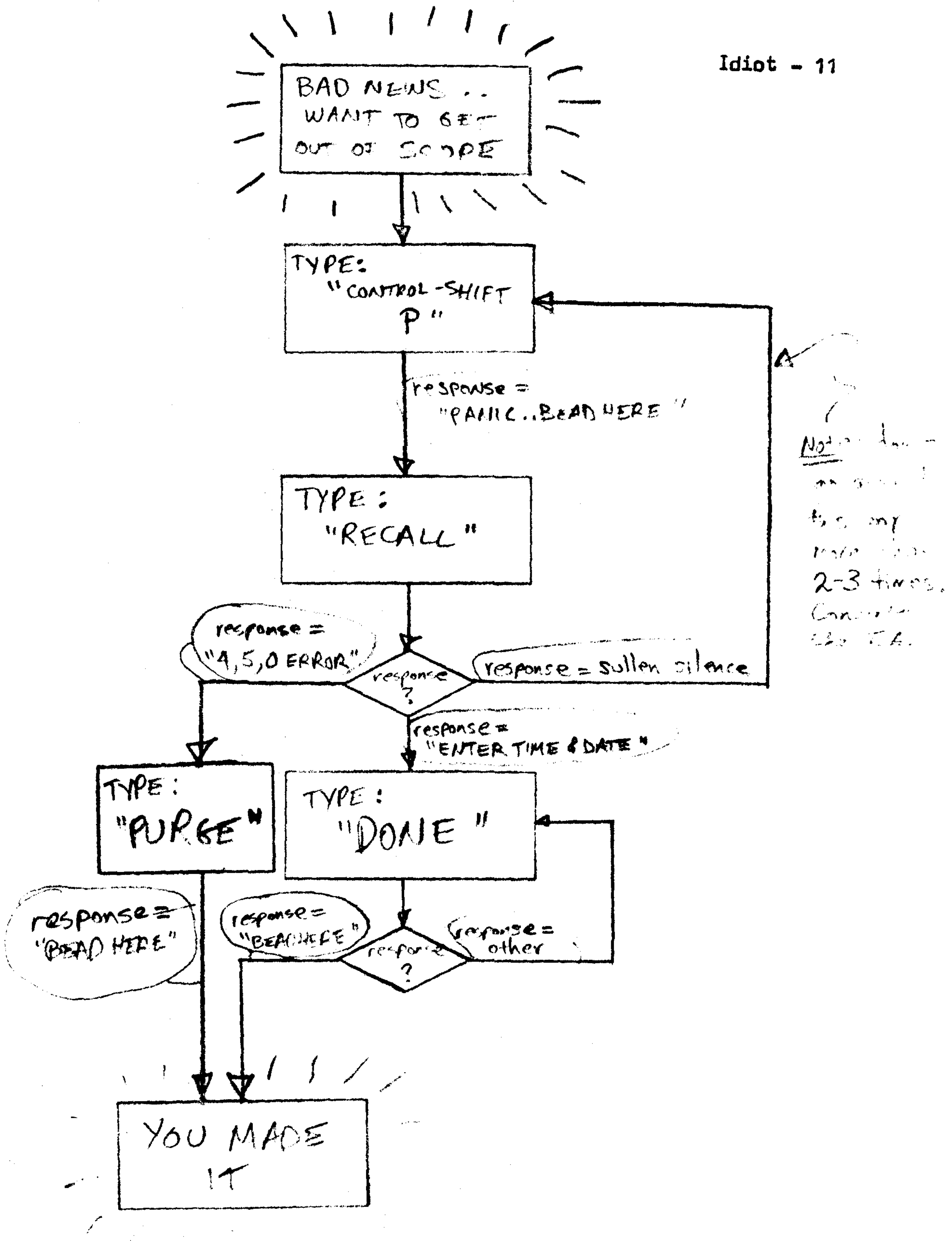
You should then type

```
RECALL
```

in an attempt to recall SCOPE. If the attempt is successful SCOPE will print

```
ENTER TIME AND DATE
```

You should then type DONE to tell SCOPE you no longer need it and to give you back to the BEAD. The above procedure will usually work. If it doesn't, consult the (rather frightening looking) flowchart on the next page. It's not nearly so bad as it looks and may get you out of trouble.


BAD NEWS .. WANT TO GET OUT OF SCOPE
TYPE: "CONTROL-SHIFT P"
response = "PANIC..BEAD HERE"
TYPE: "RECALL"
response = "4,5,0 ERROR"
response ?
response = sullen silence
response = "ENTER TIME & DATE"
TYPE: "PURGE"
TYPE: "DONE"
response = "BEAD HERE"
response = "BEAD HERE"
response ?
response = other
YOU MADE IT
2-3 times.

12. Your friend the editor.

The editor can do many things besides just inserting lines in your file and printing them out, as described earlier. It can move through the file looking for things, and can change lines or delete them entirely if you prefer. The line at which the editor is positioned is called the current line. The current line may be changed by three commands: T, M, and B. T (for top) causes the editor to position itself at the top of the file. Thus to insert something at the beginning of a file one types

```
T;I
```

The M (for Move) command has five different forms. M by itself or M1 means move one line forward (down) in the file. M7 means move seven lines forward. M.string means move forward in the file until a line containing string as a sub-part is reached. M/string means move forward in the file until a line beginning with string is reached (ignoring leading blanks). Thus . is equivalent to an unanchored search in Snobol, while / is equivalent to an anchored search (with the exception of the fact that it ignores leading blanks). If the last line in the file is reached before the particular stopping condition is met the editor continues at the first line of the file (i.e. it "wraps around"). If the condition is never met *NOT FOUND is printed and no movement occurs. Naturally, any number can replace 7 and any character string not comtaining a semicolon may replace string in the foregoing examples. Finally, M$ means move to thelast line in the file.

These five command forms occur in several of the commands described below so let us use the abbreviation sc to stand for the stop condition and say that a move command takes the form

```
Msc
```

where sc may be nothing, a number, /string, .string, or $.

Typing

```
B3
```

moves the editor back three lines in the file. B all by itself means B1. A number is the only stop condition allowed with the B command (sorry).

<u>Printing</u>. A command of the form

```
Psc
```

prints lines until the stop condition sc is met or the end of the file is reached. Specifically, P or P1 prints one line, the current one, but does not change the current line. Thus

```
P;M;P
```

is equivalent to

```
P2
```

The last line printed is the current one. To print the whole file type

```
T;P$
```

If the end of the file is reached before the stop condition is satisfied, *BOTTOM is printed and the editor stays there (unlike the move command).

<u>Deleting</u>. A command of the form

```
Dsc
```

deletes the current line together with all the lines after it up <u>through</u> the one satisfying the stop condition sc . Specifically,

```
D4
```

deletes four lines and

```
D/LOOP
```

deletes from the current line through a line beginning with LOOP . The lines you have deleted are replaced with a funny kind of line which prints as *DELETED . This line disappears as soon as the next file movement occurs. It makes the editor's life easier, so accept it.

If you want to replace a line, type

```
D;I
```

then type the replacing line(s). Don't forget to leave insert mode by typing a carriage return at the beginning of the line when your inserting is completed.

Changing. If you wish to change part of aline, rather than deleting it entirely and replacing it with another one, use the C(for change)command. This command has the form

```
C/string1/string2/sc
```

where string1 is the sequence of characters to be changed, and string2 is its replacement. Only the first instance of string1 in a line is changed. If the stop condition is omitted, the changing is done to the current line only (this is the most common case). Otherwise the first instance of string1 in each line is changed to string2 until the line satisfying the stop condition is encountered. For example, the command

```
C/LOOP1/LOOP2/4
```

asks that the first instance of LOOP1 be changed to LOOP2 in the next 4 lines, starting with the current one.

If all instances are to be replaced, rather than just the first in each line, then the change global command

```
CG/LOOP1/LOOP2/4
```

may be used. The strings represented by string1 and string2 may be of any length or may be null.

Terminating. When you're through messing around with a file you can type F to the editor and it will leave the recently edited version of your file in ECS under the name given it when the editor was initially invoked. That is, if you started with

```
C,EDITOR,S,FILE1
```

then the original version of FILE1 in ECS is lost and will be replaced by the recently edited version. If instead, you now wish to give this edited file a new name you may type

```
F,FILE2       or in general     F,fname,uname
```

This will leave FILE1 unchanged in ECS and will cause a new file, FILE2 (the modified version of FILE1) to reside there also. If you've decided that your editing was all a mistake, type

```
Q     instead of  F
```

and all your work will disappear. That is, FILE1 will remain unmodified in ECS and no new file will be generated.

13. Debugging your Snobol program.

In order to find out whatwent wrong with your Snobol program, type (when you have the BEAD's attention)

```
TRIM
SPACE,5000
C,EDITOR,S,OUTPUT
```

The editor will respond by typing EDIT as usual and will position itself at the top of your output file. If you have gotten a message saying ABORT from SCOPE (indicating compilation error(s) in Snobol) you should type

```
M.↑↑↑;B2;P
```

which will tell the editor to move to a line containing some up-arrows and then to back up two and print that line (the offending one). (Note that the output file has carriage control characters in it and will look sort of strange consequently.) You should move all through the output file looking for up-arrows so that you are sure you have found all the compiler errors before trying to execute the program again. (The M command wraps around the file, so you will eventually end up back where you started.)

If instead you had an execution error you should look for the statement number given in the error termination message. This may be done by typing

```
M.1□;P
```

if the offending statement is statement number 1. If it is not obvious why this statement should cause an execution error, do a

```
M.ERROR□;P2
```

to move to the error information which follows the program and to print two lines of information. The second line should tell you what the nature of theproblem is.

When you've located all the errors you must make changes, not in the file named OUTPUT (which is now useless) but in the file which has your Snobol program on it. You can tell the editor Q for quitting and it will return you to the BEAD. You will then have to call the editor in again by typing

```
C,EDITOR,S,fname,uname
```

where fname is the name of the file containing your Snobol program. (No need to TRIM and ask for SPACE again as the same amount of space will be needed as before.) You can now look through your Snobol program, make corrections, send the corrected program out to the disk and get it back again so it can be executed. This time when calling SCOPE there is no need to do a TEXT,OUTPUT as your output file will be generated in the proper format for the duration of your session, and every time SCOPE is called it automatically rewinds the file named OUTPUT so you are positioned at the beginning. If you have generated a very large output file whicn is no longer needed (perhaps by a program which went into an infinite loop) you can release the ECS space which it occupies by telling the BEAD

```
          K,OUTPUT        or in general        K,fname,uname
```

and it will destroy (kill) that file for you. Then a new

```
          TEXT,OUTPUT     or in general        TEXT,fname
```

under SCOPE will be necessary to put the file in a printable format.

14. Sending messages to the printer.

If your output lines are longer than 72 characters, those past 72 will be hard to read on the teletype as they will all be printed directly on top ofone another. Files containing long lines or many lines should always be printed on a printer rather than on a teletype. This is done by telling the BEAD

```
          TRIM
          SPACE,3000
          C,PRINTER,S,fname,uname
```

where fname is the name of the file to be printed.

This should evoke the response

```
          TSS PRINTER DRIVER  VER 2.0
          ENTER TITLE LINE
```

If instead the second line reads

```
          FILE NOT SYSTEM STANDARD TEXT
```

that means that you have misspelled the file name or have forgotten to do a TEXT,fname under SCOPE before you generated the file. The first case can be fixed by simply calling the printer again; the second case can be fixed by killing this bad file with a K,fname,uname and then calling SCOPE and doing a TEXT,fname. Now that the format of your output file has been specified properly, you will have to generate it all over again. (A file that has been generated by the editor is already in the proper format to be printed.)

If the printer diver thinks the file is OK its ENTER TITLE LINE request should be answered with a response of the form

jobnumber name PLEASE FILE IN THE STUDENT OUTPUT AREA

where jobnumber is the number for CS120A batch jobs and name is your name in any format that you like. At the end of the hour the operator should take the paper off the printer and file it along with the usual batch jobs for this class.

When you've entered the title, the printer will ask

SCOPE FORMAT CONVENTIONS?

This means that it wants to know whether or not your lines start with a carriage control character which the printer is to respect. If they do (all your output files will) then type Y for yes; if they don't (your input files won't) then type N for no. The printer will then say (after some delay)

EVCH CLEARED

and the keyboard ofyour teletype will become unusable while the printer is in operation. When your printing has been completed the message

BEAD HERE

will be printed and your teletype keyboard will spring back to life.

If the printer seems to be taking much longer than it should you may terminate the process with a panic (CONTROL-SHIFT-P) followed by a RECALL. This should leave you back with the BEAD.

Occasionally the printer may say

PRINTER or HARDWARE ERROR or OUT OF FORMS

or messages to that effect. This means that it is out of paper, or jammed, or in some similarly unhappy state. If this happens, ask the T.A. to notify an operator in the machine room - nothing can be printed until theprinter is made ready again mechanically.

15. Common sicknesses and their cures.

symptom 06,01,00 ERROR .. BEAD HERE

explanation: you have exhausted all the space you requested and need more

cure: type SPACE,n where n is the extra amount you think you will need in order to complete the running process. If an OK response is secured, then be sure to type

RESTORE

and everything should proceed normally from the point at which the space was exhausted. If the response is TOO MUCH , you'll have to wait or negotiate as described in Section 5.

symptom FILE fname,uname IS BUSY

explanation: too much to cope with here

cure: type CONTINUE

symptoms SAY AGAIN

??NO??

????

?

explanation: Each sub-system has a different way of indicating that it has failed to understand your command (request). The disk system says SAY AGAIN ; the BEAD says ??NO?? ; SCOPE says ???? ; the editor says simply ? . These different inquiries are helpful because they indicate to you which sub-system you are talking to, which may explain why it doesn't understand what you're trying to say.

cure: tell this subsystem something it <u>will</u> understand.

When to take a PURGE. Whenever you have encountered some sort of error from which you have not been able to do a RESTORE , then you should type

```
PURGE
```

to clean up after yourself. That should result in the appearance of the message

```
BEAD  HERE          or          OK
```

depending on what was going on. (Note the flow chart on page 11.) Under no circumstances should you ever type PURGE to the DISK as disastrous results may occur. (You should never panic out of the DISK either.)

Killing the BEAD. If the BEAD gets into a bad state where it doesn't understand a thing you say -- keeps giving you ??NO?? all the time -- then as a last resort you should ask it to commit suicide by typing

```
BYEBEAD
```

You can then cause it to be reincarnated by hitting CONTROL-SHIFT-P and everything should then be all right -- but you'll have to start over again on whatever you were in the midst of doing. Your first action should be to re-introduce yourself to this reincarnated BEAD by telling it your name, i.e. you should type

```
USER,idiot
```

again.

16. Parting is such sweet sorrow. You've probably noticed that you say farewell to each sub-system with a differen t parting word. To the disk system you say **FIN** ; the editor responds to F and Q ; SCOPE wants to be told DONE. But how does one get rid of the everpresent BEAD -- how does one tell it that you're tired and want to go in search of a strawberry milkshake, Farewell to the BEAD is expressed (unromantically) as

```
LOGOUT
```

The BEAD will respond (with equal lack of sentiment)

```
idiot  LOGGED OUT
```

You should then type

```
TRIM
```

to reduce the space used by your teletype to zero. If no one is waiting to use the teletype you may as well give the poor thing a much deserved rest by turning its left-side switch from LINE to OFF -- drink in the silence before leaving.