```
;
; ELIZA by Joseph Weizenbaum.
;
; Any line beginning with a semicolon is commentary and was not part of
; the original ELIZA code. Each commentary block generally refers to the
; code just above it.
;



        CHANGE   MAD
            EXTERNAL FUNCTION (KEY,MYTRAN)                                      000010
            NORMAL MODE IS INTEGER                                              000020
            ENTRY TO CHANGE.                                                    000030
            LIST.(INPUT)                                                        000040
            VECTOR VALUES G(1)=$TYPE$,$SUBST$,$APPEND$,$ADD$,                   000050
           1$START$,$RANK$,$DISPLA$                                             000060
            VECTOR VALUES SNUMB = $ I3 *$                                       000070
            FIT=0                                                               000080
CHANGE      PRINT COMMENT $PLEASE INSTRUCT ME$                                  001400
            LISTRD.(MTLIST.(INPUT),0)                                           001410
            JOB=POPTOP.(INPUT)                                                  001420
            THROUGH IDENT, FOR J=1,1, J.G. 7                                    001430
IDENT       WHENEVER G(J) .E. JOB, TRANSFER TO THEMA                            001440
            PRINT COMMENT $CHANGE NOT RECOGNIZED$                               001450
            TRANSFER TO CHANGE                                                  001460
THEMA       WHENEVER J .E. 5, FUNCTION RETURN IRALST.(INPUT)                    001470
            WHENEVER J .E. 7                                                    001480
                THROUGH DISPLA, FOR I=0,1, I  .G. 32                            001490
                WHENEVER LISTMT.(KEY(I)) .E. 0, TRANSFER TO DISPLA              001500
                S=SEQRDR.(KEY(I))                                               001510
READ(7)         NEXT=SEQLR.(S,F)                                                001520
                WHENEVER F .G. 0, TRANSFER TO DISPLA                            001530
                PRINT COMMENT $*$                                               001540
                TPRINT.(NEXT,0)                                                 001550
                PRINT FORMAT SNUMB,I                                            001560
                PRINT COMENT $ $                                                001570
                TRANSFER TO READ(7)                                             001580
DISPLA          CONTINUE                                                        001590
                PRINT COMMENT $ $                                               001600
                PRINT COMMENT $MEMORY LIST FOLLOWS$                             001610
                PRINT COMMENT $ $                                               001620
                THROUGH MEMLIST, FOR I=1 , 1, I .G. 4                           001630
MEMLST          TXTPRT.(MYTRAN(I),0)                                            001640
                TRANSFER TO CHANGE                                              001650
            END OF CONDITIONAL                                                  001660
            THEME=POPTOP.(INPUT)                                                001670
            SUBJECT=KEY(HASH.(THEME,5))                                         001680
            S=SEQRDR.(SUBJECT)                                                  001690
LOOK        TERM=SEQLR.(S,F)                                                    001700
            WHENEVER F .G. 0, TRANSFER TO FAIL                                  001710
            WHENEVER TOP.(TERM) .E. THEME, TRANSFER TO FOUND                    001720
            TRANSFER TO LOOK                                                    001730
FOUND       TRANSFER TO DELTA(J)                                                001740
DELTA(1)    TPRINT.(TERM,0)                                                     001750
            TRANSFER TO CHANGE                                                  001760
FAIL        PRINT COMMENT $LIST NOT FOUND$                                      001770
            TRANSFER TO CHANGE                                                  001780
DELTA(2)    S=SEQRDR.(TERM)                                                     001790
            OLD=POPTOP.(INPUT)                                                  001800
READ(1)     OBJCT=SEQLR.(S,F)                                                   001810
            WHENEVER F .G. 0, TRANSFER TO FAIL                                  001820
            WHENEVER F .NE. 0, TRANSFER TO READ(1)                              001830
            INSIDE=SEQRDR.(OBJECT)
READ(2)     IT=SEQLR.(INSIDE,F)
            WHENEVER F .G. 0, TRANSFER TO READ(1)
            SIT=SEQRDR.(IT)
            SOLD=SEQRDR.(OLD)
ITOLD       TOLD=SEQLR.(SOLD,FOLD)
            DIT=SEQLR.(SIT,FIT)
            WHENEVER TOLD .E. DIT .AND. FOLD .LE. 0,TRANSFER TO ITOLD
            WHENEVER FOLD .G. 0, TRANSFER TO OK(J)
            TRANSFER TO READ(2)
OK(2)       SUBST.(POPTOP.(INPUT),LSPNTR.(INSIDE))
            TRANSFER TO CHANGE
OK(3)       NEWBOT.(POPTOP.(INPUT),OBJCT)
            TRANSFER TO CHANGE
DELTA(3)    TRANSFER TO DELTA(2)
DELTA(4)    WHENEVER NAMTST.(BOT.(TERM)) .E. 0
                BOTTOM=POPBOT.(TERM)
                NEWBOT.(POPTOP.(INPUT),TERM)
                NEWBOT.(BOTTOM,TERM)
            OTHERWISE
                NEWBOT.(POPTOP.(INPUT),TERM)
            END OF CONDITIONAL
            TRANSFER TO CHANGE
DELTA(6)    S=SEQRDR.(TERM)
READ(6)     OBJCT=SEQLR.(S,F)
            WHENEVER F .G. 0, TRANSFER TO FAIL
            WHENEVER F .NE. 0, TRANSFER TO READ(6)
            OBJCT=SEQLL.(S,F)
            WHENEVER LNKLL.(OBJECT) .E. 0
                SUBST.(POPTOP.(INPUT),LSPNTR.(S))
            OTHERWISE
                NEWTOP.(POPTOP.(INPUT),LSPNTR.(S))
            END OF CONDITIONAL
            TRANSFER TO CHANGE
           R* * * * * * * * * * END OF MODIFICATION ROUTINE
            END OF FUNCTION
        TPRINT  MAD
            EXTERNAL FUNCTION (LST)
            NORMAL MODE IS INTEGER
            ENTRY TO TPRINT.
            SA=SEQRDR.(LST)
            LIST.(OUT)
READ        NEXT=SEQLR.(SA,FA)
            WHENEVER FA .G. 0, TRANSFER TO P
            WHENEVER FA .E. 0, TRANSFER TO B
            POINT=NEWBOT.(NEXT,OUT)
            WHENEVER SA .L. 0, MRKNEG.(POINT)
            TRANSFER TO READ
B           TXTPRT.(OUT,0)
            SEQLL.(SA,FA)
MORE        NEXT=SEQLR.(SA,FA)
            WHENEVER TOP.(NEXT) .E. $=$
                TXTPRT.(NEXT,0)
                TRANSFER TO MORE
            END OF CONDITIONAL
            WHENEVER FA .G. 0, TRANSFER TO DONE
            PRINT COMMENT $ $
            SB=SEQRDR.(NEXT)
MEHR        TERM=SEQLR.(SB,FB)
            WHENEVER FB .L.0
                PRINT ON LINE FORMAT NUMBER, TERM
                VECTOR VALUES NUMBER = $I3 *$
                TRANSFER TO MEHR
            END OF CONDITIONAL                                                  000280
            WHENEVER FB .G. 0, TRANSFER TO MORE                                 000290
            TXTPRT.(TERM,0)                                                     000300
            TRANSFER TO MEHR                                                    000310
P           TXTPRT.(OUT,0)                                                      000320
DONE        IRALST.(OUT)                                                        000330
            FUNCTION RETURN                                                     000340
            END OF FUNCTION                                                     000350
        LPRINT  MAD
            EXTERNAL FUNCTION (LST,TAPE)                                        006340
            NORMAL MODE IS INTEGER                                              006350
            ENTRY TO LPRINT.                                                    006360
            BLANK = $      $                                                    006370
            EXECUTE PLACE.(TAPE,0)                                              006380
            LEFTP = 606074606060K                                               006390
            RIGHTP= 606034606060K                                               006400
            BOTH  = 607460603460K                                               006410
            EXECUTE NEWTOP.(SEQRDR.(LST),LIST.(STACK))                          006420
            S=POPTOP.(STACK)                                                    006430
BEGIN       EXECUTE PLACE.(LEFTP,1)                                             006440
NEXT        WORD=SEQLR.(S,FLAG)                                                 006450
            WHENEVER FLAG .L. 0                                                 006460
            EXECUTE PLACE.(WORD,1)                                              006470
            WHENEVER S .G. 0, PLACE.(BLANK,1)                                   006480
            TRANSFER TO NEXT                                                    006490
            OR WHENEVER FLAG .G. 0                                              006500
            EXECUTE PLACE.(RIGHTP,1)                                            006510
            WHENEVER LISTMT.(STACK) .E. 0, TRANSFER TO DONE                     006520
            S=POPTOP.(STACK)                                                    006530
            TRANSFER TO NEXT                                                    006540
            OTHERWISE                                                           006550
            WHENEVER LISTMT.(WORD) .E. 0                                        006560
            EXECUTE PLACE.(BOTH,1)                                              006570
            TRANSFER TO NEXT                                                    006580
            OTHERWISE                                                           006590
            EXECUTE NEWTOP.(S,STACK)                                            006600
            S=SEQRDR.(WORD)                                                     006610
            TRANSFER TO BEGIN                                                   006620
            END OF CONDITIONAL                                                  006630
            END OF CONDITIONAL                                                  006640
DONE        EXECUTE PLACE.(0,-1)                                                006650
            EXECUTE IRALST.(STACK)                                              006660
            FUNCTION RETURN LST                                                 006670
            END OF FUNCTION                                                     006680
;
; TESTS(CAND, S) return a sequence reader if the keyword matches the user's
;                input text, otherwise return 0.
;
; CAND  is the keyword candidate transformation rule
; S     is the sequence reader for the user INPUT text
;
; This function has 3 tasks
;
;  1. Test whether the whole candidate keyword matches the whole word
;     in the user's input text.
;  2. If the words do match, make any keyword substitution specified
;     in the candidate transformation rule.
;  3. Position the candidate reader past the substitution keyword, if any.
;
; SLIP packs 6 6-bit characters into each 36-bit IBM 7094 machine word.
; If a word has more than 6 characters it is continued into the next SLIP
; cell, with the first cell having its sign bit set. ???
;
```

```
; This code abstracts this full-word matching and has the side-effect
; of modifying the user's input text with the substitution word, if
; specified.
;
        TESTS   MAD
            EXTERNAL FUNCTION(CAND,S)                                           000010
            NORMAL MODE IS INTEGER                                              000020
            DIMENSION FIRST(5),SECOND(5)                                        000030
            ENTRY TO TESTS.                                                     000040
            STORE=S                                                             000050
            READER=SEQRDR.(CAND)                                                000060
            THROUGH ONE, FOR I=0,1, I .G. 100                                   000070
            FIRST(I)=SEQLR.(READER,FR)                                          000080
ONE         WHENEVER READER .G. 0, TRANSFER TO ENDONE                           000090
;
; Copy all 6-character chunks of the candidate keyword to the FIRST array.
;
; [As the loop termination condition is I .G. 100 (000070), this code will
; write past the end of the FIRST array if the keyword is longer than 36
; characters (because the first 36 characters will be copied to
; FIRST(0) .. FIRST(5), and any further characters will be written to
; machine words past FIRST(5)).]
;
ENDONE      SEQLL.(S,F)                                                         000100
            THROUGH TWO, FOR J=0,1, J .G. 100                                   000110
            SECOND(J)=SEQLR.(S,F)                                               000120
TWO         WHENEVER S .G. 0, TRANSFER TO ENDTWO                                000130
;
; Copy all 6-character chunks of the user input word to the SECOND array.
; [May write past the end of SECOND.]
;
ENDTWO      WHENEVER I .NE. J, FUNCTION RETURN 0                                000140
;
; If the keyword in FIRST has a different number of 6-character chunks to
; the word in SECOND the two words cannot be the same, so return the value 0,
; signifying no match.
;
; WHENEVER is an abbreviation of WHENEVER
; .NE. means not equal
; FUNCTION RETURN is an abbreviation of FUNCTION RETURN
;
            THROUGH LOOK, FOR K=0,1, K.G. J                                     000150
LOOK        WHENEVER FIRST(K) .NE. SECOND(K), FUNCTION RETURN 0                 000170
;
; Compare each 6-character chunk of the keyword with the corresponding chunk
; of the user input word. If any are different, return 0, signifying no match.
;
            EQL=SEQLR.(READER,FR)                                               000180
            WHENEVER EQL .NE. $=$                                               000190
            SEQLL.(READER,FR)                                                   000200
            FUNCTION RETURN READER                                              000210
            OTHERWISE                                                           000220
;
; At this point we know that the keyword matches the user's word.
; Check whether the transformation rules specify a simple word substitution,
; signified by the presence of an "=".
;
; If it is not an "=", reposition the reader back before the element and
; return the reader, signifying a successful match.
;
            POINT=LNKL.(STORE)                                                  000230
            THROUGH DELETE , FOR K=0,1, K .G. J                                 000240
            REMOVE.(LSPNTR.(STORE))                                             000250
```

```
DELETE      SEQLR.(STORE,F)                                                     000260
INSRT       NEW=SEQLR.(READER,FR)                                               000270
            POINT=NEWTOP.(NEW,POINT)                                            000280
            MRKNEG.(POINT)                                                      000290
            WHENEVER READER .L. 0, TRANSFER TO INSRT                            000300
            MRKPOS.(POINT)                                                      000310
            FUNCTION RETURN READER                                              000320
            END OF CONDITIONAL                                                  000330
            END OF FUNCTION                                                     000340
;
; An "=" was present in the transformation rule. E.g. a script
; transformation rule may begin
;
;           (YOUR = MY
;               ((0 MY 0)
;                   (WHY ARE YOU CONCERNED OVER MY 3)
;                   (WHAT ABOUT YOUR OWN 3)
;                   :
;
; Say at this point the keyword YOUR has been found in the user's input text
; and we know that in the transformation rule the keyword (YOUR) is followed
; by an "=". So we're now going to replace the YOUR in the input text with
; the word following the "=" in the transformation rule (MY, in this case).
;
; First delete all the 6-character chunks that comprise this word, then
; insert all the 6-character chunks that comprise the replacement word.
;
; Finally, return the reader, signifying a successful match.
;
        DOCBCD  MAD
            EXTERNAL FUNCTION (A,B)                                             000010
            NORMAL MODE IS INTEGER                                              000020
            ENTRY TO FRBCD.                                                     000030
            WHENEVER LNKL.(A) .E. 0, TRANSFER TO NUMBER                         000040
            B=A                                                                 000050
            FUNCTION RETURN 0                                                   000060
NUMBER      K=A*262144                                                          000070
            B=BCDIT.(K)                                                         000080
            FUNCTION RETURN 0                                                   000090
            END OF FUNCTION                                                     000100
;
; ELIZA entry point.
;
        ELIZA   MAD
            NORMAL MODE IS INTEGER                                              000010
            DIMENSION KEY(32),MYTRAN(4)                                         000020
;
; KEY     - A hashmap used to record keywords.
;           KEY(0)..KEY(31)  is the keyword->transformation rule hashmap
;           KEY(32)          is the "NONE" transformation rule
;
; MYTRAN  - A hashmap used to record the MEMORY rules.
;           MYTRAN(1)..MYTRAN(4) contain the four MEMORY rules.
;
; A note on MAD arrays: DIMENSION D(N) allocates N+1 machine-words of
; core memory, which are accessed using indexes 0..N.
;
            INITAS.(0)                                                          000030
;
; INITAS must be the first executable statement in any program using SLIP.
; Its purpose is to create the List of Available Space from all unused
; core memory. It does not require an argument, but here is given 0.
;
```

```
            PRINT COMMENT $WHICH SCRIPT DO YOU WISH TO PLAY$                    000060
            READ FORMAT SNUMB,SCRIPT                                            000070
;
; Display the message "WHICH SCRIPT DO YOU WISH TO PLAY".
;
; Note that the IBM 7090/7094 character set doesn't include a question
; mark glyph. Also $ is used to delimit character strings.
;
; SNUMB is the FORTRAN format string " I3 *", defined previously, which
; expects the user to enter up to 3 decimal digits. This number is assigned
; to the variable SCRIPT and will be used as the tape drive unit number
; where the ELIZA script is expected to reside.
;
            LIST.(TEST)                                                         000080
            LIST.(INPUT)                                                        000090
            LIST.(OUTPUT)                                                       000100
            LIST.(JUNK)                                                         000110
;
; Initialise four lists. These are:
; TEST    - Used to store the parts of the user's text matching a
;           decomposition rule.
; INPUT   - During ELIZA startup the selected script is read into this list,
;           one round-bracketed list at a time.
;           During the conversation phase the text entered by the user is
;           read into this list.
; OUTPUT  - ELIZA's response sentence is constructed in this list.
; JUNK    - A list used for temporary storage for several different purposes.
;
            LIMIT=1                                                             000120
;
; When Weizenbaum talks in the January 1966 CACM paper of a "certain counting
; mechanism", it is this to which he is referring. LIMIT has the value 1..4,
; in order, and then restarts at 1. The value changes to the next in the
; sequence at each user input. More on LIMIT below.
;
            LSSCPY.(THREAD.(INPUT,SCRIPT),JUNK)                                 000130
            MTLIST.(INPUT)                                                      000140
;
; The THREAD function reads text from the tape unit specified by the integer
; SCRIPT into the INPUT list. The LSSCPY function copies the first list in
; that INPUT to the list named JUNK.
;
; The first list in an ELIZA script must be the hello message, e.g.
; (HOW DO YOU DO.  PLEASE TELL ME YOUR PROBLEM).)
;
            THROUGH MLST, FOR I=1,1, I .G. 4                                    000150
MLST        LIST.(MYTRAN(I))                                                    000160
;
; Initialise each of the four MYTRAN array entries as a new list.
;
; THROUGH is an abbreviation for THROUGH
; .G. is the Boolean grater than operator
;
; Set I to 1, if I is greater than 4 stop looping, otherwise execute the code
; up to and including the statement labelled MLST. Then add 1 to I and return
; to the top of the loop at the point of the test to see if I is greater than
; 4 and repeat.
;
;   for I in 1..4 {
;     call function LIST with argument a reference
;       to the Ith entry in the MYTRAN array
;   }
;
```

```
            MINE=0                                                          000170
            LIST.(MYLIST)                                                   000180
;
; MINE    - Set to 0 and is never changed. It's referenced once below. ???
; MYLIST  - As memories are made using the MYTRAN MEMORY rules they are
;           recorded in MYLIST. Here MYLIST is being initialised as a new
;           empty list.
;
            THROUGH KEYLST, FOR I=0,1, I .G. 32                             000220
KEYLST      LIST.(KEY(I))                                                   000230
;
; Initialise each of KEY(0) .. KEY(32) array entries as a new list.
;   for I in 0..32 {
;     call function LIST with argument a reference
;       to the Ith entry in the KEY array
;   }
;
           R* * * * * * * * * * READ NEW SCRIPT                             000240
BEGIN       MTLIST.(INPUT)                                                  000250
            NODLST.(INPUT)                                                  000260
            LISTRD.(INPUT,SCRIPT)                                           000270
;
; Empty the INPUT list. Remove the description list??? from INPUT (NODLST).
; Read the next round-bracket-delimited list from tape unit id SCRIPT.
;
            WHENEVER LISTMT.(INPUT) .E. 0                                   000280
                TXTPRT.(JUNK,0)                                             000290
                MTLIST.(JUNK)                                               000300
                TRANSFER TO START                                           000310
            END OF CONDITIONAL                                              000320
;
; WHENEVER is an abbreviation of WHENEVER.
; .E. means equals.
; TRANSFER TO is an abbreviation for TRANSFER TO.
; END OF CONDITIONAL is an abbreviation of END OF CONDITIONAL.
;
; An empty list signals the end of the ELIZA script. (Which is presumably
; why there is () on the last line of the published DOCTOR script.)
;
;   if INPUT is the empty list {
;     (the whole ELIZA script has now been read and processed)
;     print the value of JUNK, e.g. "HOW DO YOU DO.  PLEASE TELL ME YOUR PROBLEM"
;     clear the JUNK list
;     goto the START label
;   }
;
            WHENEVER TOP.(INPUT) .E. $NONE$                                 000330
                NEWTOP.(LSSCPY.(INPUT,LIST.(9)),KEY(32))                    000340
                TRANSFER TO BEGIN                                           000350
;
; If this list is the special "NONE" list, just copy it unchanged into KEY(32)
; and then goto BEGIN to read the next list in the script.
;
; Recall that the NONE list in the DOCTOR script is:
;       (NONE
;           ((0)
;               (I AM NOT SURE I UNDERSTAND YOU FULLY)
;               (PLEASE GO ON)
;               (WHAT DOES THAT SUGGEST TO YOU)
;               (DO YOU FEEL STRONGLY ABOUT DISCUSSING SUCH THINGS)))
;
               OR WHENEVER TOP.(INPUT) .E. $MEMORY$                         000360
                POPTOP.(INPUT)                                              000370
                MEMORY=POPTOP.(INPUT)                                           000380
                THROUGH MEM, FOR I=1,1, I .G. 4                                 000390
MEM             LSSCPY.(POPTOP.(INPUT),MYTRAN(I))                               000400
                TRANSFER TO BEGIN                                               000410
;
; Otherwise, if this list is the special "MEMORY" list, process it into the
; four MYTRAN lists. Recall that the MEMORY list looks like this and is
; required to have exactly four transformation patterns:
;       (MEMORY MY
;           (0 YOUR 0 = LETS DISCUSS FURTHER WHY YOUR 3)
;           (0 YOUR 0 = EARLIER YOU SAID YOUR 3)
;           (0 YOUR 0 = BUT YOUR 3)
;           (0 YOUR 0 = DOES THAT HAVE ANYTHING TO DO WITH THE FACT THAT YOUR 3))
;
;   else if the first word in INPUT is "MEMORY" {
;     assign the memory keyword (e.g. "MY") to the MEMORY variable
;     for I in 1..4 {
;       copy the Ith MEMORY pattern/reconstruction to MYTRAN(I)
;     }
;     goto the BEGIN label (continue reading the ELIZA script)
;   }
;
               OTHERWISE                                                        000420
                NEWBOT.(LSSCPY.(INPUT,LIST.(9)),KEY(HASH.                       000430
           1    (TOP.(INPUT),5)))                                               000440
                TRANSFER TO BEGIN                                               000450
            END OF CONDITIONAL                                                  000460
;
; Otherwise, the first word in the INPUT list is expected to be a keyword.
; Insert this keyword into the KEY hashtable, so that
;   KEY(HASH(keyword)) -> list of transformation rules for keywords
;                         that hash to this entry in KEY (i.e. more than
;                         one keyword may hash to the same entry in KEY,
;                         so each entry in KEY may have zero, one or many
;                         keyword transformation rules associated with it.)
;
; (1 in column 11 signifies a continuation of the previous line.)
;
; The HASH function takes a word and a number (N) and returns a deterministic
; value between 0 and (2 to the power N)-1, in this case 0..31.
;
;   else {
;     HASH the keyword and append this transformation rule to the
;       entry in KEY with that index
;     goto the BEGIN label (continue reading the ELIZA script)
;   }
;
; This is the end of the script reading code. When the script has been
; read and processed the script reader explicitly jumps to the START label
; to begin the user conversation.
;
           R* * * * * * * * * * BEGIN MAJOR LOOP                                000470
START       TREAD.(MTLIST.(INPUT),0)                                            000480
;
; Wait for the user to type a sentence and read it into the INPUT list,
; which is first cleared. Presumably, tape unit 0 is the console.
;
; TREAD is the SLIP system text read function.
;
            KEYWRD=0                                                            000490
            PREDNC=0                                                            000500
;
; KEYWRD  - This will be the keyword found to have the highest precedence.
; PREDNC  - The precedence of the keyword. Precedence is specified in the
;           ELIZA script. E.g. (DREAMS = DREAM 3 (=DREAM)), the keyword
;           DREAMS is given the precedence value 3.
;
            LIMIT=LIMIT+1                                                       000510
            WHENEVER LIMIT .E. 5, LIMIT=1                                       000520
;
; Increment the value of LIMIT. If it then equals 5, set it back to 1.
; If we just read the very first user input, LIMIT will now have the value 2.
;
            WHENEVER LISTMT.(INPUT) .E. 0, TRANSFER TO ENDPLA                   000530
;
; If the user input is a blank line, goto the ENDPLA label.
; A blank user input tells ELIZA the conversation is over.
;
            IT=0                                                                000540
;
; IT      - On exit from the scanning loop IT will either be the sequence
;           reader for the selected transformation rule, or it will be 0
;           indicating that no keyword was detected in the user's INPUT.
;
            WHENEVER TOP.(INPUT) .E. $+$                                        000550
                CHANGE.(KEY,MYTRAN)                                             000560
                TRANSFER TO START                                               000570
            END OF CONDITIONAL                                                  000580
;
; If first word of the user input is a "+" character, call the CHANGE
; function defined higher up in this code. This function allows the user
; to modify the current ELIZA script with the commands TYPE, SUBST,
; APPEND, ADD, START, RANK and DISPLA.
; After making any changes, return to the START label and carry on the
; conversation.
;
            WHENEVER TOP.(INPUT) .E. $*$, TRANSFER TO NEWLST                    000590
;
; If first word of the user input is a "*" character, goto the NEWLST label.
; NEWLST is defined later in this code. It inserts a new transformation rule,
; which the user will have given after the "*", into the current in-memory
; script and then returns to the START label to carry on the conversation.
;
            S=SEQRDR.(INPUT)                                                    000600
;
; Create the Slip sequence reader, S, for the user's INPUT list.
;
NOTYET      WHENEVER S .L. 0                                                    000610
                SEQLR.(S,F)                                                     000620
                TRANSFER TO NOTYET                                              000630
;
; ???
;
               OTHERWISE                                                        000640
                WORD=SEQLR.(S,F)                                                000650
                WHENEVER WORD .E. $.$ .OR. WORD .E. $,$ .OR. WORD .E. $BUT$     000660
                    WHENEVER IT .E. 0                                           000670
                        NULSTL.(INPUT,LSPNTR.(S),JUNK)                          000680
                        MTLIST.(JUNK)                                           000690
                        TRANSFER TO NOTYET                                      000700
                       OTHERWISE                                                000710
                        NULSTR.(INPUT,LSPNTR.(S),JUNK)                          000720
                        MTLIST.(JUNK)                                           000730
                        TRANSFER TO ENDTXT                                      000740
                       END OF CONDITIONAL                                       000750
                    END OF CONDITIONAL                                          000760
                END OF CONDITIONAL                                              000770
;
; Set the variable WORD to the next word in the user's INPUT list. Then
; test that word to see if it's a delimiter.
;
; Note that in Weizenbaum's 1966 CACM paper, only comma and period were
; listed as delimiters. And yet the example conversation given in that
; paper could not be reproduced unless BUT is also a delimiter.
;
; Note that WORD is a 36-bit integer. Weizenbaum developed ELIZA between
; 1964 and 1966 on an IBM 7094, which has a 36-bit word and uses a 6-bit
; character encoding. Characters were packed 6 to a word. In Slip, character
; strings longer than six characters are stored in successive list cells.
; In this case WORD=SEQLR.(S,F) is assigning the first six characters of
; the next word in the user's INPUT text to the integer variable WORD.
; If the word had fewer than six characters they would be left justified
; with space characters padding to the right.
;
;
;   else {
;     if WORD is one of the delimiters ".", "," or "BUT" {
;       if we have found no keywords in the INPUT so far (IT .E. 0) {
;         discard all words in INPUT to the left of, and including, this
;           delimiter
;         goto NOTYET and continue scanning the rest of the user INPUT
;           for keywords
;       }
;       else {
;         discard all words in INPUT to the right of, and including, this
;           delimiter
;         goto ENDTXT; scanning of the user INPUT is now complete
;       }
;     }
;   }
;
                WHENEVER F .G. 0, TRANSFER TO ENDTXT                            000780
;
; If there were no more words to read in the user INPUT list, goto the
; ENDTXT label; scanning of the user INPUT is now complete.
; (F will be 1 when the sequence reader has traversed the whole INPUT
; list and is back at the list header.)
;
                I=HASH.(WORD,5)                                                 000790
                SCANER=SEQRDR.(KEY(I))                                          000800
                SF=0                                                            000810
                THROUGH SEARCH, FOR J=0,0, SF .G. 0                             000820
                CAND= SEQLR.(SCANER,SF)                                         000830
                WHENEVER SF .G. 0, TRANSFER TO NOTYET                           000840
SEARCH          WHENEVER TOP.(CAND) .E. WORD, TRANSFER TO KEYFND                000850
;
; Is WORD a keyword? Try to locate it in the KEY hashmap.
;
; Recall that more than one keyword may hash to the same entry in KEY,
; so each entry in KEY is a list that may have zero, one or many keyword
; transformation rules associated with it. We need to look through this
; list to see if it contains a keyword that exactly matches WORD.
;
;   HASH the WORD to get the index I in the KEY table where this word
;     would have been stored, if it is a keyword
;   loop {
;     try to read the next candidate list from the hashmap entry KEY(I)
;     if there isn't another candidate list {
;       WORD didn't match any entries so it's not a keyword
;       goto NOTYET to continue scanning the user's input text
;     }
;     if WORD is the same as the first entry in this candidate list {
;       WORD is a keyword and CAND is the transformation rule for
;       this keyword, so goto KEYFND
;     }
;   }
;
KEYFND          READER=TESTS.(CAND,S)                                           000860
                WHENEVER READER .E. 0, TRANSFER TO NOTYET                       000870
;
; Call the TESTS function, defined higher up in this code.
;
; TESTS checks that the whole keyword matches the whole user INPUT word. It
; also performs any keyword substitution in the user INPUT. (e.g. (MY = YOUR))
;
; If TESTS returns 0 it means the keyword is not identical to the word in
; the user input, so goto NOTYET to continue scanning the user INPUT.
;
; [This suggests that keywords must differ in the first six characters.
; (Because TESTS is called only for the first keyword candidate in
; the KEY hashmap that matches the first six characters of the user's
; input word).]
;
                WHENEVER LSTNAM.(CAND) .NE. 0                                   000880
                    DL=LSTNAM.(CAND)                                            000890
SEQ                 WHENEVER S .L. 0                                            000900
                        SEQLR.(S,F)                                             000910
                        TRANSFER TO SEQ                                         000920
                       OTHERWISE                                                000930
                        NEWTOP.(DL,LSPNTR.(S))                                  000940
                    END OF CONDITIONAL                                          000950
                   OTHERWISE                                                    000960
                END OF CONDITIONAL                                              000970
;
; ???
;
                NEXT=SEQLR.(READER,FR)                                          000980
                WHENEVER FR .G. 0, TRANSFER TO NOTYET                           000990
;
; Read the next element in the rules associated with this keyword.
; If we are back at the rules header, the rules list was empty, so goto
; NOTYET to continue scanning the user INPUT.
;
                WHENEVER IT .E. 0 .AND. FR .E. 0                                001000
PLCKEY              IT=READER                                                   001010
                    KEYWRD=WORD                                                 001020
;
; 001000 If this is the first keyword we've encountered in the user's INPUT
; (IT .E. 0), and the first entry in the associated rules is a list
; rather than a value (FR .E. 0)???, i.e. there is no precedence associated
; with this keyword, then record the associated rules reader in IT and
; the found keyword in KEYWRD. Then goto NOTYET (001100) to continue
; scanning the user's input.
;
                   OR WHENEVER FR .L. 0 .AND. NEXT .G. PREDNC                   001030
                    PREDNC=NEXT                                                 001040
                    NEXT=SEQLR.(READER,FR)                                      001050
                    TRANSFER TO PLCKEY                                          001060
                   OTHERWISE                                                    001070
                    TRANSFER TO NOTYET                                          001080
                END OF CONDITIONAL                                              001090
                TRANSFER TO NOTYET                                              001100
```

```
               R* * * * * * * * * * END OF MAJOR LOOP                           001110
;
; 001030 Otherwise, if the first entry in the associated rules is a value???
; (FR .L. 0), i.e. the precedence of this keyword, and that value is greater
; than the precedence of the previously found highest precedence keyword
; (NEXT .G. PREDNC), then record the new highest precedence value in PREDNC
; and move the rule reader past the precedence value, then goto PLCKEY to
; also record the reader in IT and the found keyword in KEYWRD. Finally, goto
; NOTYET (001100) to continue scanning the user's input.
;
; Note that this differs from Weizenbaum's CACM paper, where it says that
; keywords of higher precedence are added to the top of a keyword stack and
; keywords of lower precedence are added to the bottom of this stack. This
; also means this code does not support the "NEWKEY" functionality he
; describes.
;
; [Note that this code implies that keywords in the script should never
; specify a precedence value of 0. If they do they would never be used
; (because NEXT .G. PREDNC will never be true).]
;
; 001080 Otherwise, ignore this keyword and return to NOTYET to continue
; scanning the user's INPUT.
;
ENDTXT          WHENEVER IT .E. 0                                               001120
                    WHENEVER LIMIT .E. 4 .AND. LISTMT.(MYLIST) .NE. 0           001130
                        OUT=POPTOP.(MYLIST)                                     001140
                        TXTPRT.(OUT,0)                                          001150
                        IRALST.(OUT)                                            001160
                        TRANSFER TO START                                       001170
                       OTHERWISE                                                001180
                        ES=BOT.(TOP.(KEY(32)))                                  001190
                        TRANSFER TO TRY                                         001200
                    END OF CONDITIONAL                                          001210
;
; 001120 If IT is 0 it means we did not find any keywords in the user's
; input, so we cannot construct a response from the user's input combined
; with any of the transformation rules in the script.
;
; Instead we do one of two things: either print one of the memories we
; previously recorded in MYLIST, if any, or we use one of the messages
; from the NONE list (which is recorded in KEY(32)).
;
; 001130 This is the mysterious "when a certain counting mechanism is in a
; particular state": recall a memory only if the memory list (MYLIST) isn't
; empty and LIMIT happens to have the value 4.
;
                   OR WHENEVER KEYWRD .E. MEMORY                                001220
                    I=HASH.(BOT.(INPUT),2)+1                                    001230
                    NEWBOT.(REGEL.(MYTRAN(I),INPUT,LIST.(MINE)),MYLIST)         001240
                    SEQLL.(IT,FR)                                               001250
                    TRANSFER TO MATCH                                           001260
;
; Otherwise, we did find a keyword (IT .E. 0 is false). If the keyword is
; the MEMORY keyword ("MY" in the DOCTOR script), then add a new memory to
; MYLIST before we carry on processing the transformation rules associated
; with the matched keyword.
;
; In the 1966 CACM paper, Weizenbaum says the selection of one of the
; transformations on the MEMORY list is random. The code shows that the
; selection is determined by the HASH value of the last word in the user's
; input. This means ELIZA conversations are repeatable, not random. If we
; have the HASH algorithm we should be able to reproduce the exact
; conversation. (The HASH function is part of the SLIP system.)
```

```
;
                   OTHERWISE                                                    001270
                    SEQLL.(IT,FR)                                               001280
;
; Otherwise, the keyword we found isn't the MEMORY keyword, so just position
; the transformation rule sequence reader past the keyword and fall through
; to the matching code.
;
               R* * * * * * * * * * MATCHING ROUTINE                            001290
MATCH               ES=SEQLR.(IT,FR)                                            001300
                    WHENEVER TOP.(ES) .E. $=$                                   001310
                        S=SEQRDR.(ES)                                           001320
                        SEQLR.(S,F)                                             001330
                        WORD=SEQLR.(S,F)                                        001340
                        I=HASH.(WORD,5)                                         001350
                        SCANER=SEQRDR.(KEY(I))                                  001360
SCAN                    ITS=SEQLR.(SCANER,F)                                    001370
                        WHENEVER F .G. 0, TRANSFER TO NOMATCH(LIMIT)            001380
                        WHENEVER WORD .E. TOP.(ITS)                             001390
                            S=SEQRDR.(ITS)                                      001400
SCANI                       ES=SEQLR.(S,F)                                      001410
                            WHENEVER F .NE.0, TRANSFER TO SCANI                 001420
                            IT=S                                                001430
                            TRANSFER TO TRY                                     001440
                        OTHERWISE                                               001450
                            TRANSFER TO SCAN                                    001460
                        END OF CONDITIONAL                                      001470
                    END OF CONDITIONAL                                          001480
                    WHENEVER FR .G. 0, TRANSFER TO NOMATCH(LIMIT)               001490
;
; If this keyword is a link to another keyword, switch to that keyword.
;
; An ELIZA script rule may have the form (HOW (=WHAT)). If the keyword
; HOW appears in the user's input and this transformation rule is selected,
; ELIZA will use the transformation rule associated with the keyword WHAT
; to generate its response.
;
;   read the next decomposition rule from the selected transformation rule
;   if the decomposition rule starts with an "=" symbol {
;     assign the word after the "=" to WORD
;     lookup WORD in the KEY hashmap
;     if WORD doesn't exist in the KEY hashmap {
;       (presumably this indicates a logical inconsistency in the script)
;       goto one of the NOMATCH(1) .. NOMATCH(4) labels to print
;         a message such as "HMMM" and back to the main conversation loop
;       which NOMATCH label is selected is determined by the value LIMIT
;         happens to have at this time
;     }
;     else {
;       position IT at first decomposition rule for the linked keyword
;       goto the TRY label to try to apply the decomposition rule
;     }
;   }
;   else if there were no (or no more) decomposition rules (FR .G. 0) {
;     (does this indicate an incorrectly formed script?)
;     goto one of the NOMATCH(1) .. NOMATCH(4) labels
;   }
;
TRY                 WHENEVER YMATCH.(TOP.(ES),INPUT,MTLIST.(TEST)) .E. 0,TRANSFER TO MATCH
001500
;
; Attempt to match the current decomposition rule (TOP.(ES)) to the user's
; INPUT.
```

```
;
; If it doesn't match (YMATCH returns 0), goto MATCH to try the next
; decomposition rule in the current transformation rule set.
;
; If it does match, the list TEST will contain the decomposed matching parts
; of the INPUT text ready for reassembly. E.g. ???
;
; The YMATCH function is part of the SLIP system.
;
                    ESRDR=SEQRDR.(ES)                                            001510
                    SEQLR.(ESRDR,ESF)                                            001520
                    POINT=SEQLR.(ESRDR,ESF)                                      001530
                    POINTR=LSPNTR.(ESRDR)                                        001540
                    WHENEVER ESF .E. 0                                           001550
                        NEWBOT.(1,POINTR)                                        001560
                        TRANS=POINT                                              001570
                        TRANSFER TO HIT                                          001580
                       OTHERWISE                                                 001590
                        THROUGH FNDHIT,FOR I=0,1, I .G. POINT                    001600
FNDHIT                  TRANS=SEQLR.(ESRDR,ESF)                                  001610
                        WHENEVER ESF .G. 0                                       001620
                            SEQLR.(ESRDR,ESF)                                    001630
                            SEQLR.(ESRDR,ESF)                                    001640
                            TRANS=SEQLR.(ESRDR,ESF)                              001650
                            SUBST.(1,POINTR)                                     001660
                            TRANSFER TO HIT                                      001670
                           OTHERWISE                                             001680
                            SUBST.(POINT+1,POINTR)                               001690
                            TRANSFER TO HIT                                      001700
                        END OF CONDITIONAL                                       001710
                    END OF CONDITIONAL                                           001720
;
; Select one of the reassembly rules associated with this decomposition rule.
;
; Reassembly rules are used in turn. This code adds a counter (001560) to
; the rules and uses it to record which reassembly rule was used last (001690).
; When all reassembly rules have been used (001620) the counter is returned
; to 1 (001660) and the first rule is used again.
;
HIT                 TXTPRT.(ASSMBL.(TRANS,TEST,MTLIST.(OUTPUT)),0)               001730
                    TRANSFER TO START                                            001740
                END OF CONDITIONAL                                               001750
;
; Finally, use the selected reassembly rule (TRANS) and list of decomposition
; parts (TEST) to assemble a response (in the list OUTPUT) and print it. Then
; goto the START label to await the next user input and continue the
; conversation.
;
; The ASSMBL function is part of the SLIP system.
;
; The END OF CONDITIONAL (END OF CONDITIONAL) on line 001750 closes the OTHERWISE
(OTHERWISE)
; on line 001270. ???
;
               R* * * * * * * * * * INSERT NEW KEYWORD LIST                      001760
NEWLST          POPTOP.(INPUT)                                                   001770
                NEWBOT.(LSSCPY.(INPUT,LIST.(9)),KEY(HASH.                        001780
               1(TOP.(INPUT),5)))                                                001790
                TRANSFER TO START                                                001800
               R* * * * * * * * * * DUMP REVISED SCRIPT                          001810
ENDPLA          PRINT COMMENT $WHAT IS TO BE THE NUMBER OF THE NEW SCRIPT$       001820
                READ FORMAT SNUMB,SCRIPT                                         001830
                LPRINT.(INPUT,SCRIPT)                                            001840
                NEWTOP.(MEMORY,MTLIST.(OUTPUT))                                 001850
                NEWTOP.($MEMORY$,OUTPUT)                                        001860
                THROUGH DUMP, FOR I=1,1 I .G. 4                                 001870
DUMP            NEWBOT.(MYTRAN(I),OUTPUT)                                       001880
                LPRINT.(OUTPUT,SCRIPT)                                          001890
                MTLIST.(OUTPUT)                                                 001900
                THROUGH WRITE, FOR I=0,1, I .G. 32                              001910
POPMOR          WHENEVER LISTMT.(KEY(I)) .E. 0, TRANSFER TO WRITE               001920
                LPRINT.(POPTOP.(KEY(I)),SCRIPT)                                 001930
                TRANSFER TO POPMOR                                              001940
WRITE           CONTINUE                                                        001950
                LPRINT.(MTLIST.(INPUT),SCRIPT)                                  001960
                EXIT.                                                           001970
               R* * * * * * * * * * SCRIPT ERROR EXIT                           001980
NOMATCH(1)      PRINT COMMENT $PLEASE CONTINUE $                                002200
                TRANSFER TO START                                               002210
NOMATCH(2)      PRINT COMMENT $HMMM $                                           002220
                TRANSFER TO START                                               002230
NOMATCH(3)      PRINT COMMENT $GO ON , PLEASE $                                 002240
                TRANSFER TO START                                               002250
NOMATCH(4)      PRINT COMMENT $I SEE $                                          002260
                TRANSFER TO START                                               002270
                VECTOR VALUES SNUMB= $I3 * $                                    002280
                END OF PROGRAM                                                  002290
```