Text Editor With Dictionary Facility

Thesis submitted in partial fulfilment of the requirements for the award of the Degree of

MASTER OF TECHNOLOGY

IN

COMPUTER SCIENCE

K. DINESH GEORGE

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES
JAWAHARLAL NEHRU UNIVERSITY
NEW DELHI - 110067
DECEMBER 1990

CERTIFICATE

This is to certify that the thesis entitled **" TEXT EDITOR WITH DICTIONARY FACILITY "**, being submitted by me to Jawaharlal Nehru University in partial fulfilment of the requirement for the award of the degree of Master of Technology is a record of original work done by me under the supervision of Dr. P.C. Saxena, Associate professor, School of Computer and System Sciences during the Monsoon semester, 1990.

The results reported in this thesis have not been submitted partially or fully to any other University or Institution for the awarding of any degree etc.

(K. DINESH GEORGE)

Prof. N.P. Mukherjee,
Dean,
School of Computer and
Systems Sciences,
JNU, New Delhi.

Dr. P.C. Saxena,
Associate Professor,
School of Computer and
Systems Sciences,
JNU, New Delhi.

ACKNOWLEDGEMENTS

First of all I am highly indebted to my supervisor Dr. P.C. Saxena, Associate Professor, School of Computer and Systems Sciences, Jawaharlal Nehru University, for his constant guidance and whelming encouragement throughout my thesis work.

I am very grateful to Prof. N.P. Mukherjee for allowing me to carry out the thesis work at the Centre itself.

I am also very much indebted to all teaching and non-teaching staff in the School of Computer and Systems Sciences, without whose help I would have not been able to complete my thesis work.

CONTENTS

CHAPTER 1

INTRODUCTION

The main modes of man-machine communication is by three different means

1. Typing
2. Writing
3. Speaking.

Actually man-machine communication is dominated by typing. The advantage of typing is that machines can sense which key is pressed with 100% accuracy. But it is a relatively slow form of communication. Writing is a more universal skill than typing. But it is considerably slow and the major disadvantage is machine recognition. Speech is potentiallly the faster form of man-machine communication. Speaking rates vary from about 120-250 words per minute. Being the most natural form of human communication, it is seriously considered in the human use of computers.

It is useful to be able to describe spoken utterances by means of discrete symbols representing the sounds that have been produced. A convenient unit for representing speech sounds is the phoneme. The set of phonemes used in each language may be different. A consistent set of phonetic symbols for languages is provided by the International Phonetic Alphabet. Any utterance can be represented by a string of symbols called as transcription of the utterance. A phonemic transcription consists of a string of phonemes.

One of the major areas in which a lot of work has been done is in automatic speech recognition (Ainsworth, 1988). There are a number of actual and potential applications of automatic speech recognition. The main current use of computers in offices are for word processing and database applications. It has long been a dream of designers of speech-recognitions systems to produce a machine which could type any utterance spoken into it. Such a machine would revolutionize offices. But it requires a continuous speech recognizer with an unlimited vocabulary.

Meisel (1986) has argued that it is practical to design a system with considerably less than 100% recognition accuracy. The output of which could be corrected by keyboard or speech editing. Meisel argues that an error rate of 5% or greater would be acceptable for a speech-to-text system (Meisel, 1984) and such a system can be produced.

A fundemental requirement of a speech-to-text system is a large vocabulary. The 1000 most frequent words make up about 75% of a typical text, but about 5000 words are required to cover 95%. The other 5% depend very much on the user, the subject matter and style. Thus truly a very large vocabulary of the order of 100,000 words is required. One solution is to provide a user specific vocabulary in which the number of words will be very limited and hence smaller vocabulary is needed (Kurzweil, 1985).

Bahl et al (1984) has done considerable research in this area and they developed a system used for typing business letters, which can operate with a vocabulary of 5000 words.

My major area of work has been to create a text editor with dictionary facility. The dictionary used will pertain to a particular user. As the vocabulary of a user is quite limited the dictionary size will be small. The idea of providing a dictionary is to use it in automatic generation of text, which could be further extended to speech recognition. That if the first or first few characters of a word are given the whole word could be generated . Similarly if the first word is given the next word can be generated by using the dictionary. A successful attempt to make a completely working system has been made by use of a test algorithm. The implementation has been done in two parts .

1. Text Editor
2. Dictionary facility

Both these are explained in detail in Chp.2 and Chp.3 respectively.

CHAPTER 2

TEXT EDITOR

2.1 Introduction

There are many editors or word processors available in the commmercial market with a variety of facilities which makes the end-users job more easy. They provide a variety of editing functions like cursor movements , scroll functions , line editing functions and text saving and printing functions.

For my project a need was there to provide its own editor. The editor provides mostly all of the important editing functions provided by any commmercial word processsor.

2.2 Functions provided by the editor

The various functions provided by the editor can be vaguely classified as follows :

1. Cursor movement and Scroll functions.
2. Line editing functions.
3. Saving and Printing functions.

These functions can be used by using the keys on the cursor keypad and <CTRL> key. One of the main feature of the editor is that it provides auto-wrap facility. That is when the number of characters in a line exceeds 80 characters then the last word will be entered on the next line and the previous line will be margin adjusted by entering extra blanks in between the words, such that the 80th character on the line will be last character of a word or any glyphs.

2.2.1. Cursor movement and Scroll functions

The functions under this group are mentioned and explained as follows :

a> Cursor left (<-) : Cursor is moved one column to the left from the current cursor position.

b> Cursor right (->) : Cursor is moved one column to the right from the current cursor position.

c> Cursor up (|) : Cursor is moved one row above the the current cursor position.

d> Cursor down (|) : Cursor is moved one row below the current cursor position.

e> Beginning of line (HOME) : Cursor is moved to the first column in the current line.

f> End of line (END) : Cursor is moved to the last character on the current line.

g> Beginning of the page (CTRL_HOME) : Cursor is moved to the first column of the first line of the current page.

h> End of page (CTRL_END) : Cursor is moved to the last character of the last line of the current page.

i> Page down (PG DN) : It is a scroll function which displays next 23 lines of text from the current cursor position.

j> Page up (PG UP) : It is a scroll function which displays previous 23 lines of text from the current cursor position.

k> Beginning of file (CTRL_PG UP) : It is a scroll function which positions the cursor at the beginning of the text being edited.

l> End of file (CTRL_PG DN) : It is a scroll function which positions the cursor at the end of the text being edited.

2.2.2 Line editing functions

The various line editing functions are mentioned and explained below.

a> Insert/Overwrite characters (INSERT) : It is a toggle switch. When INSERT is on then characters are inserted before the current cursor position in the current line. When OVERWRITE is on then the characters which are typed are written at the cursor position itself.

b> Delete character (DELETE/BACK SPACE) : If one presses DELETE key , then the character at the cursor position is deleted. If one presses the BACK SPACE key then the character left to the cursor position is deleted.

c> Delete a line (CTRL_Y) : It deletes the current line being edited from the text.

d> Insert a line (CTRL_N) : It inserts a blank line above the current cursor position.

e> Create a line (ENTER) : It creates a new blank line below the current cursor position.

2.2.3. Saving and Printing functions

They allow online and offline operations. In online operation i.e while editing a text, the function keys provide the required function.

(F1) Save the current text being edited on to disk.

(F4) Print the current text being edited.

In offline operation the Main Menu provides the options for printing or saving a file.

2.3 Data structures used in Text Editor program

In keeping in view of all the functions mentioned above, provided by the Text Editor an efficient data structure had to be used, such that functions like deleting a line, inserting a line, cursor movements, scrolling etc can easily be implemented.

The data structure used in my project and its representation is shown in fig. 1 of Appendix A. The data types used for the editor programs are :

```
TYPE
     LPTR = ^NODE;

     NODE = RECORD
               LLINK : LPTR;
                LINE : STRING[80];
               RLINK : LPTR
            END;
```

As we can see the data structure used is a doubly linked list with each node having three fields. Two fields contain the left and right link information pointing to other nodes in the list and the third field can store a string of 80 characters. This has been done keeping in view the fact that , normally a CRT screen can display 80 characters in a line.

The advantage of having a doubly linked list is that one can move through the list to the left or to the right from any node position. This facilitates easy cursor function implementation. Also deleting a line is implemented by deleting a node from the list and creating a line is done by creating a new node and inserting into the linked list.

CHAPTER 3

DICTIONARY FACILITY

3.1 Introduction

One of the major parts of the project is the inbuilt dictionaries. There are four dictionaries.

1. Two single word dictionaries
2. One double word dictionary.
3. One triple word dictionary.

It has been found that most of the users vocabulary especially for typing documents or letters is limited, and also that some double word and triple word phrases are very often used by an user. Taking this into acccount we have four dictionaries as mentioned above for each user which can be used in his computing environment. The dictonary for each user is different because the vocabulary and the common phrases used by different users may be different.

Each users dictionary is created by using an already existing set of documents written by the user. The program takes each of the documents one by one and parses them into single words , double words and triple words combinations and stores them in the dictionary with their frequency information . The frequency information gives the frequency of occurence of the words and their combinations in the documents used to create the dictionary.

The program also provides on-line updation of the single word dictionaries. That is if a word being typed is not there in the dictionary it is automatically entered into it.

3.2 DATA STRUCTURES

One of the major design aspect of creating a dictionary is deciding on which data structure to use and how to store and retrieve information in an efficient way. Various data structures that can be used and various strorage and retrieval methods that can be used are described in Horowitz et al (1984).

The data structure used in my project and its representation is shown in fig. 2 of appendix A. As an example the data types used for single word dictionaries are :

```
TYPE
      SWPTR = ^SWNODE;

      SWNODE = RECORD
                  LLINK : SWPTR;
                  WORD  : STRING[25];
                  FREQ  : INTEGER;
                  RLINK : SWPTR
               END;

      FSWNODE = RECORD
                  COUNT : INTEGER;
                  NEXT  : SWPTR;
                END;

      SWDICT = ARRAY ['A'..'Z'] OF FSWNODE;
```

As represented, there are 26 buckets, each corresponding to each character in the alphabet set. Each bucket has two fields. One for the link to the list of nodes containing the words and the other containing the number of nodes in the list. Each node in the list contains four fields . Two fields contain the left and right link information pointing to other nodes in the doubly linked list. One field contains the word and the other field contains the frequency of occurance of the word in the documents used for creating the dictionary.

As mentioned there are four dictionaries. For creating the dictionaries we parse the input documents into singly linked list of words with period information, which is a flag information which specifies the end of a sentence. The period information helps us to find all double word and triple word combinations till the sentence delimiter. The representation is shown in fig. 3 of appendix A with an example. The data types used for this purpose is given below :

```
TYPE
     PTR2 = ^WORDNODE;

     WORDNODE = RECORD
                     PRD    : BOOLEAN;
                     DATA   : STRING[25];
                     RIGHT  : PTR2
                END;
```

This data structure is a singly linked list with each node having three fields. One field is a link information to the next node in the list. The DATA field contains the single words and PRD is a boolean field which is true if the charater after the word in the sentence is any sentence delimiter like ('.',':','?','!',...). This field information will be useful for forming double word and triple word combinations.

Example of a piece of text and how double words and triple words are formed and what information is stored is given below :

eg. I have a faithful friend. Do you have a friend?

SINGLE WORD	FREQUENCY
A	2
DO	1
FAITHFUL	1
FRIEND	2
HAVE	2
I	1
YOU	1

DOUBLE WORD	FREQUENCY
A FAITHFUL	1
A FRIEND	1
DO YOU	1
FAITHFUL FRIEND	1
HAVE A	2
I HAVE	1
YOU HAVE	1

TRIPLE WORD	FREQUENCY
A FAITHFUL FRIEND	1
DO YOU HAVE	1
HAVE A FAITHFUL	1
HAVE A FRIEND	1
I HAVE A	1
YOU HAVE A	1

As can be made out from the data types, hashing is used to store and retrieve information from the data structures. There are twenty-six buckets (see fig.2) as mentioned earlier and each word or words combination is hashed into any of these buckets depending on the first character of the word or of the words combination. The overflow handling is achieved by using chaining.

As the single words or double or triple words combinations are hashed, it is checked whether they exist in the chain of the bucket or not. If they exist then their frequency of occurence is incremented, else a new node is created and entered in the chain with frequency as one.

As mentioned earlier there are two single word dictionaries. In one, the chains are ordered according to the frequency of occurence of the single words corresponding to each bucket. In the other single word dictionary the chains are arranged in lexicographic order. In the double and triple word dictionaries the chains are arranged in lexicographic order.

In double and triple word dictionaries when the words combination is entered, it is made sure that there is no more than one blank character between the words.

3.3 USING THE EDITOR WITH DICTIONARY FACILITY

The editor program can be used in two modes.

1. Normal editing (NORMAL MODE).
2. Editing with dictionary facility (PROMPT MODE).

One can switch from one mode to the other by F10 function key. The normal editing mode has already been explained in chapter 2.

The prompt mode helps the user to use the inbuilt dictionary for editing. This mode helps the user in reducing the number of keystrokes for typing a document.

The test algorithm used is given in section 3.4 and is explained in short below.

In prompt mode , a prompt is displayed to input the first character of a word to be typed. Given the first character (x), the program displays first five words of highest frequencies starting with the input character (x) from the single word dictionary in which the words are arranged according to frequency. The user can select any of the first five words

displayed else he can type the next character of the word (y) he wants. Then the program displays five words starting with characters (xy), if it exists, according to the frequency of occurence of the words. If it does not exist the user has to type the whole word. This new word will be automatically entered into the single word dictionaries, thus allowing on-line updation of dictionary. This new word can be used later without typing all the characters of the word.

If the word was selected from the display then the program uses this single word and goes to the double word dictionary and dislays the first five double words starting with the single word selected earlier. One may select a double word from this display or he may type the first character of the second word of the double word he wants. If so the program displays five double words, if it exists, with second word starting with the character typed. If such double words do not exist then the program goes to single word dictionary and loops again.

If a double word has been selected then the program goes to the triple word dictionary to display the first five triple words starting with the double word selected before. The user may select any of these or he may type the first character of the third word he wants in the triple word. Using this information the program again goes to the triple word dictionary to display triple words accordingly, if it exists. Else it goes to double word dictionary and loops.

If we select a triple word from the display the program takes the double word combination of second and third word and goes to the triple word dictionary and loops.

Note that this is only a test algorithm. One can modify this algorithm in many ways and run on a test data and compare the effiency of the various algorithms. The main test is to find which algorithm provides the best efficiency in typing. That is which algorithm when used , forces one to make the least number of key strokes.

3.4 TEST ALGORITHM FOR FORMING A SENTENCE

```
Begin ;

read D; {Display prompt and read D as first character of the word
        to be typed }

display 5 single words in order of frequency from single word
          dictionary starting with character D;

read SCH; { read the number on the display or any character }

if SCH in ('1'..'5') then TSWRD contains the selected single word

else
  {
    if SCH <> ' ' then SCHARS := D + SCH

    Do single word linear search; { search single word dictionary
                                    for words starting with SCHARS;}
    if found then
              {
                  read(SCH);

                  if SCH in ('1'..'5') then TSWRD contains single
                    word selected from display;

                  else
                    {
                      SCHARS := SCHARS + SCH;

                      goto Do single word linear search;
                      ----
                    }
              }
```

```
    else
                {
                    read(SCH);

                    if SCH is not any delimiter character then
                       {
                         SCHARS := SCHARS + SCH; { keep on adding
                                                   the  new character typed
                                              until it is a delimiter one }

                         goto Do single word linear search;
                         ----
                       }
                    else
                       {
                         update single word dictionary; {with the
                                                    new word typed in}

                         goto Begin;
                         ----
                       }
                }
   }

SWRD-D := '';

DCHARS := TSWRD + ' ';

Do double word linear search; { search double word dictionary for
                                double words starting with DCHARS}

if found  then
          {
              read (SCH);

              if  SCH  in  ('1'..'5')  then  DWRD  contains  the
                    selected double word;
              else
                  {
                      DCHARS := DCHARS + SCH;

                      SWRD-D := SWRD-D + SCH;

                      goto Do double word linear search;
                      ----
                  }
          }
```

```
 SWRD-D = '' then goto Begin;
                  ----
 SWRD-D = a single character then
        {
           D := SWRD-D;
           goto Begin; { and dont read D}
           ----
        }
se
        {
           SCHARS := SWRD-D;

           goto Do single word linear search;
           ----
        }

SWRD2 from DWRD; { SWRD1 contains the first word
                   and SWRD2 contains the second
                   word of the double word in
                   DWRD}


 ' ';

linear search;{ search triple word dictionary for
                triple words starting with TCHARS}


ead(SCH);

f SCH in ('1'..'5') then TWRD contains the
  selected triple word;
lse
   {
       TCHARS := TCHARS + SCH;

       SWRD3 := SWRD3 +SCH;

       goto Do triple word linear search;
   }



CHARS := SWRD2 + ' ' + SWRD3;

oto Do double word linear search;
---
```

```
find DWRD from TWRD ; {DWRD will contain last twowords from the
                            triple word in TWRD}

goto Do triple word linear search;
----
End.
```

CHAPTER 4

RESULTS, ANALYSIS AND CONCLUSION

The Text editor has been successfully implemented and a test algorithm for automatically generating sentences and text using the dictionary facility has also been successfully implemented. The dictionary facility is very useful in writing documents. The main reason is that the vocabulary of an user is limited and if we use a collection of documents written by the same user, one can create the dictionaries and their size will be small. Also since the user may use same phrases or sentences in his documents, especially letters, the dictionary will really help the user to type a document very fast as most of the phrases already exist in the dictionary and even if the user is slow in typing he can do the typing job at a faster rate as he has to make very few key strokes. The efficiency in using the dictionary facility of the program may be calculated as follows :

$$\text{Efficiency}(\%) = 100 - \frac{\text{Actual key strokes made by user}}{\text{Total no of characters in the text}} \times 100$$

If the efficiency is 80% it means that out of 100 characters in the text only 20 characters had to be typed in by the user actually . The rest of the 80 characters have been added by the use of the dictionary.

The principle used in this project can be used in **automatic speech recognition** (Ainsworth, 1988). As we know speech is formed by concatenating various phonemes. If we have an inbuilt dictionary of various words pronounced with their phonemes and its concatenation , then just by using first few phonemes the whole word can be recognised by the speech recognizer (Bahl, 1984). This has been a major area of research in fifth generation computers where speech synthesis and recognition forms a major area of research (Miller, 1987).

The algorithms used in the program are not the most efficient ones. Also a lot of redundancy can be avoided in coding as well as in storage of dictionary. In my project all the dictionaries are memory resident at the time of editing. When the dictionaries are large the whole of dictionary cannot be loaded into memory all at the same time. Hence efficient memory management techniques have to be used so as to reduce the number of read or writes from or into the disks, and also to reduce the memory requirements and increase the program execution speed.

Different algorithms can tried other than the test algorithm used here in chapter 3, for using the dictionaries in forming sentences. We can do a comparitive study and analysis of different algorithms in editing the same piece of text by them.

Different suggestions of algorithms that can be tried are given below.

1. One may not create or use the triple word dictionary at all in the program.

2. One may have all the dictionaries but we can use different automatic sentence forming algorithms other than the one used here and compare their efficiencies. Different heuristics can be used in forming the algorithms.

Also study can be made of automatically generating a text by using the highest frequency words and words combinations only and making a study of how many words which were automatically generated were correctly used.

BIBLIOGRAPHY

AINSWORTH, W.A.(1988) : Speech recognition by machine (Peter Pereguins Ltd., London)

BAHL, L.R.(1984) : ' Some experiments with large vocabulary isolated-word sentence recognition'. Proc IEEE ICASSP

KURZWEIL, R. (1985) : ' The kurzweil voicewriter : a large vocabulary voice activated word processor'. Proc Speech Tech'85 (Media Dimension Inc.)

MEISEL, W.S. (1984) : 'Speech-to-text systems the users needs.' Prcc. 1st Int. Conf. Speech Technology, Brighton.

MEISEL, W.S. (1980) : 'Towards the "talk writer" ' in BRISTOW, G. electronic speech recognition (Colins, London)

MILLER, RICHARD. K. (1987) : 'Fifth generation computers' (The Fairmount Press, Inc.)

Appendix 'A'

Figures

DOUBLY LINKED LIST RERESENTATION FOR STORING TEXT LINE BY LINE

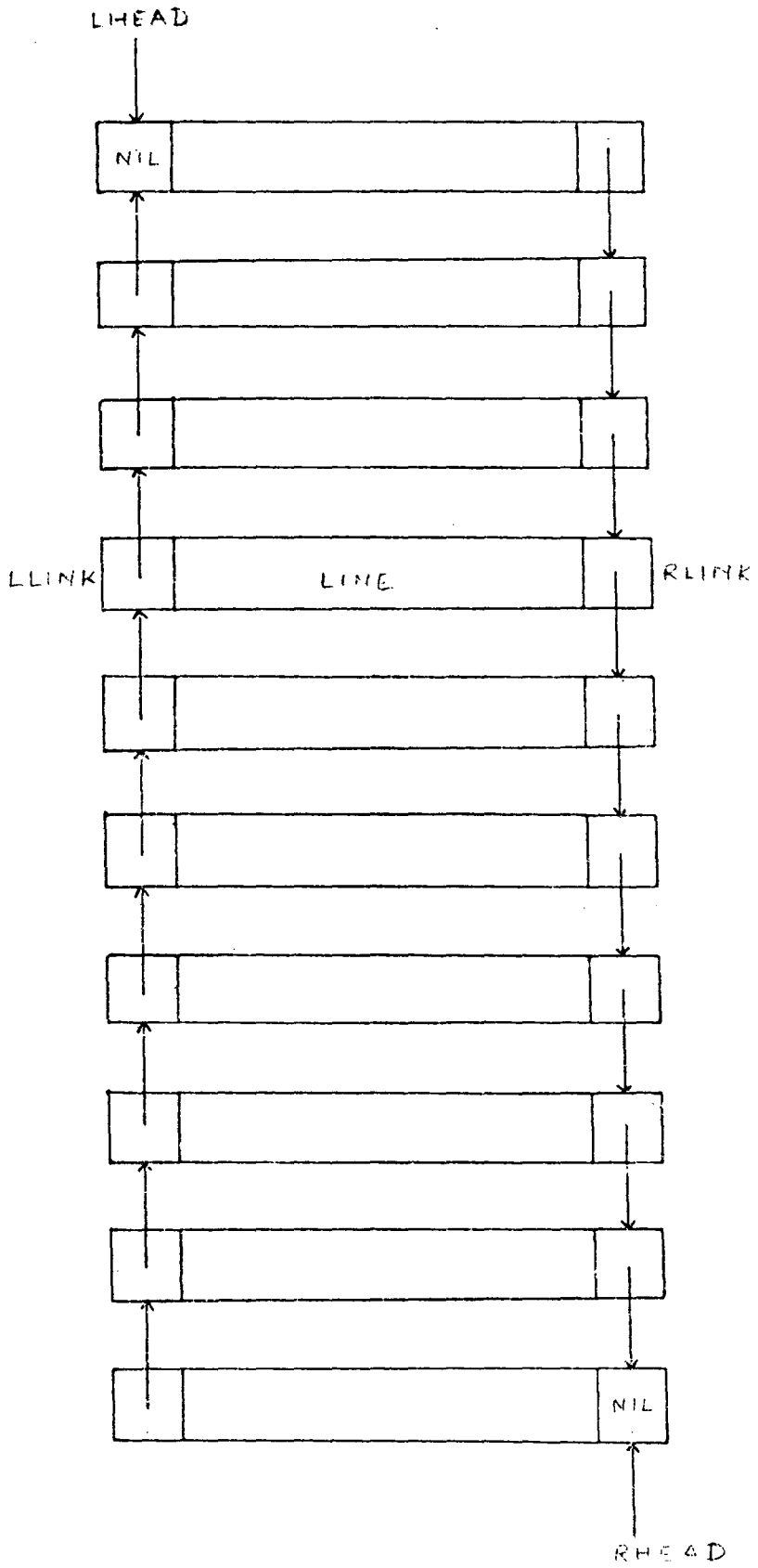

FIG 1

REPRESENTATION OF SINGLE WORD DICTIONARY

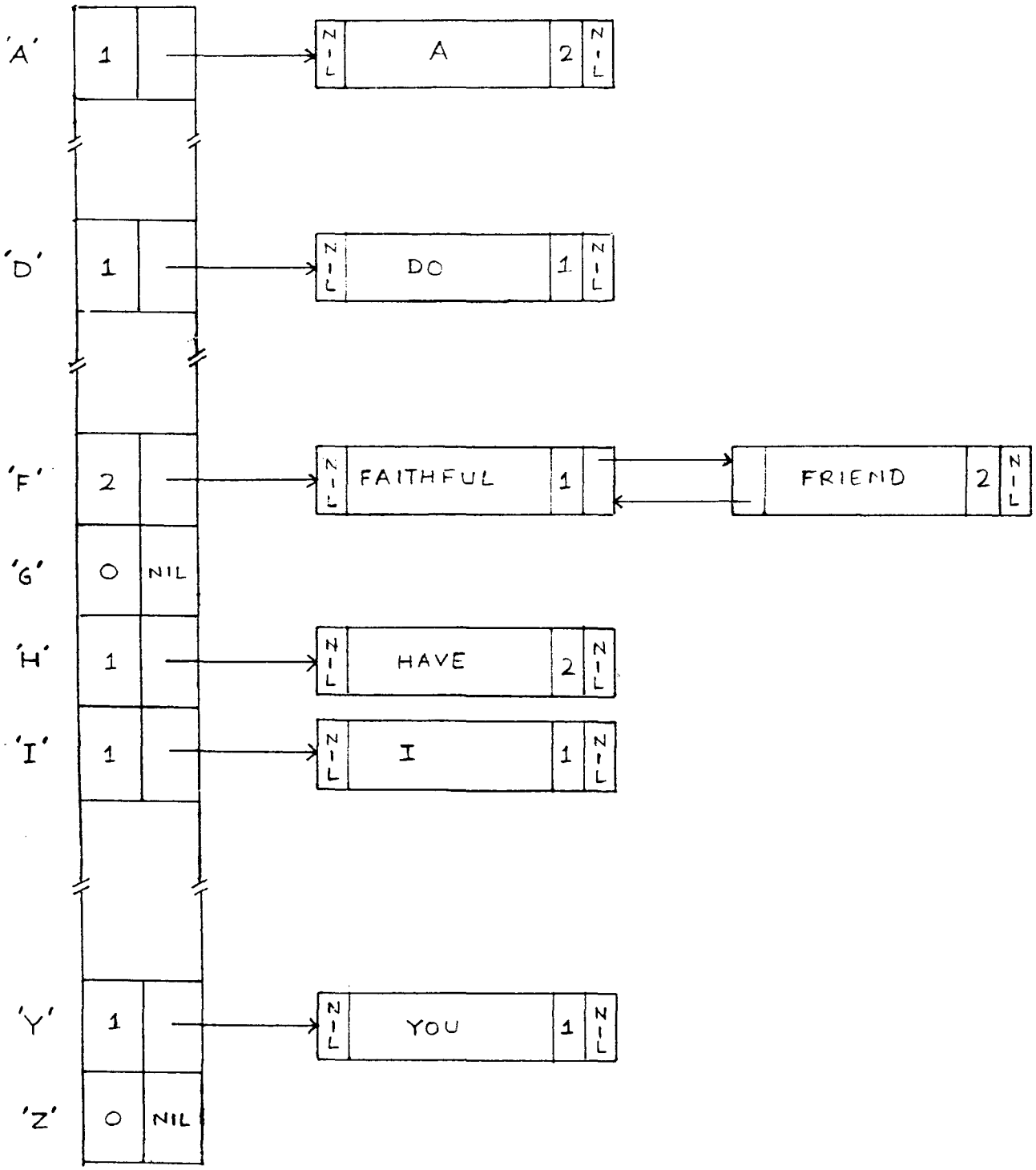

FIG 2

SINGLY LINKED LIST OF WORDS WITH PERIOD INFORMATION

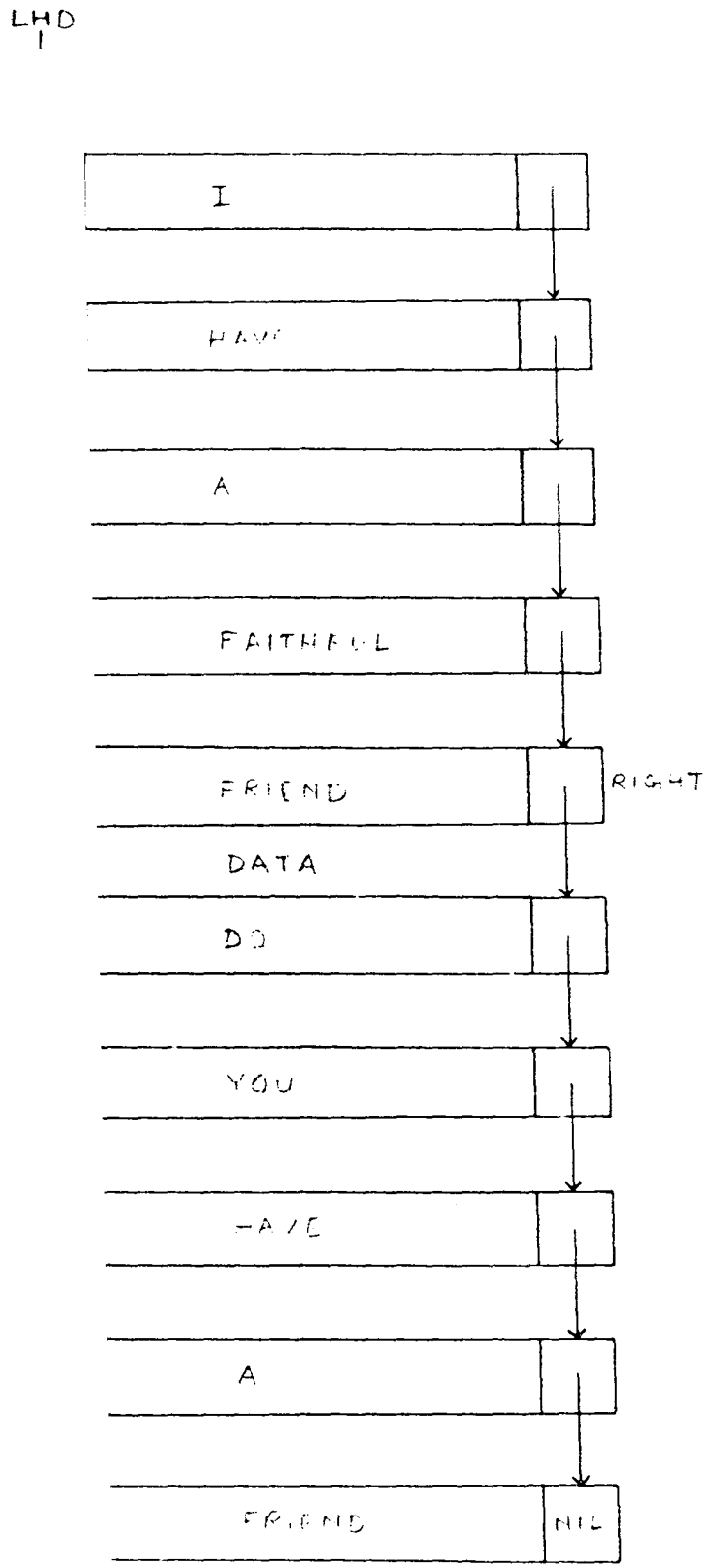

FIG 3

Appendix 'B'

Program Source Listing

```
PROGRAM DICTMAKE(INPUT,OUTPUT);
LABEL TT1,T1,L10,L11;
CONST MAX = 50;
TYPE
     RECTYPE =(RCNT,ACTREC);
     SWREC = RECORD
             CASE RECKIND :RECTYPE OF
                  RCNT:(NOREC :INTEGER);
                  ACTREC:(WORD:STRING[25];
                          FREQ:INTEGER;)
             END;

     DWREC = RECORD
             CASE RECKIND :RECTYPE OF
                  RCNT:(NOREC :INTEGER);
                  ACTREC:(WORD:STRING[30];
                          FREQ:INTEGER;)
             END;

     TWREC = RECORD
             CASE RECKIND :RECTYPE OF
                  RCNT:(NOREC :INTEGER);
                  ACTREC:(WORD:STRING[40];
                          FREQ:INTEGER;)
             END;

    SWPTR = ^SWNODE;
    SWNODE = RECORD
             LLINK:SWPTR;
             WORD:STRING[25];
             FREQ:INTEGER;
             RLINK:SWPTR
           END;
    FSWNODE = RECORD
               COUNT :INTEGER;
                NEXT: SWPTR;
             END;

    DWPTR = ^DWNODE;
    DWNODE = RECORD
             LLINK:DWPTR;
             WORD:STRING[30];
             FREQ:INTEGER;
             RLINK:DWPTR
           END;
    FDWNODE = RECORD
               COUNT :INTEGER;
                NEXT: DWPTR;
             END;
```

```
    TWPTR = ^TWNODE;
    TWNODE = RECORD
              LLINK:TWPTR;
              WORD:STRING[40];
              FREQ:INTEGER;
              RLINK:TWPTR
             END;
    FTWNODE = RECORD
               COUNT :INTEGER;
                NEXT: TWPTR;
              END;

    PTR2 = ^WORDNODE;
    WORDNODE = RECORD
               PRD:BOOLEAN;
               DATA:STRING[25];
               RIGHT:PTR2;
               END;

    SREC = RECORD
            WORD:STRING[25];
            FREQ:INTEGER;
           END;

    DREC = RECORD
            WORD:STRING[30];
            FREQ:INTEGER;
           END;

    TREC = RECORD
            WORD:STRING[40];
            FREQ:INTEGER;
           END;

    SWDICT = ARRAY ['A'..'Z'] OF FSWNODE;
    DWDICT = ARRAY['A'..'Z'] OF FDWNODE;
    TWDICT = ARRAY['A'..'Z'] OF FTWNODE;
    WRD1 = STRING[25];
    WRD2 = STRING[30];
    WRD3 = STRING[40];

    LPTR = ^NODE;
     NODE = RECORD
        LLINK : LPTR;
        LINE: STRING[80];
        RLINK: LPTR;
            END;

VAR  LHEAD ,RHEAD,P,Q,R,X,Y,Z: LPTR;
     TXTFN:STRING[20];
     A,B:STRING[80];
     TXTFV:TEXT[2048];
     I,J,K,ROW,COL,TROW:INTEGER;
     FIRST,F:BOOLEAN;
```

```
    BLARR:ARRAY [1..30] OF INTEGER;
    SWDIC,SWDIC1:SWDICT; DWDIC:DWDICT;TWDIC:TWDICT;
    FNAME,FNAME2:STRING[20];
    FV:TEXT[2048];
    M,LPOS:INTEGER;
    PRMT,EDTR,FLAG1,QUIT_EDIT,FLAG2,FLAG,
          PERIOD,FOUND,DONE,SATIS:BOOLEAN;
    CI,CH,C,D: CHAR;
    LEN,LNO,KEYCNT,INDEX,FREQ,TMP,VAL,COUNT: INTEGER;
    SWNO,TWNO,DWNO:INTEGER;
    SQ1,SP,SQ,SZ,SX:SWPTR;
    DP,DQ,DZ,DX,DQ1:DWPTR;
    TP,TQ,TZ,TX,TQ1:TWPTR;
    P2,LHD,Q2,R2,S2 : PTR2;
    TSWRD,SWRD,SWRD1,SWRD2,SWRD3:WRD1;
    DWRD,DWRD1,DWRD2:WRD2;
    TWRD,TWRD1,TWRD2:WRD3;
    SFV,S1FV:FILE OF SWREC;
    DFV:FILE OF DWREC;
    TFV:FILE OF TWREC;
    SWR,SWR1:SWREC;
    DWR:DWREC;
    TWR:TWREC;
    SWARR,SWARR1,SWARR2,SWARR3:ARRAY [1..MAX] OF SREC;
    DWARR,DWARR2,DWARR3:ARRAY [1..MAX] OF DREC;
    TWARR,TWARR2,TWARR3:ARRAY [1..MAX] OF TREC;
    SR,TSR:SREC;DR,TDR:DREC;TR,TTR:TREC;
    SCHARS:WRD1;DCHARS,DCHARST:WRD2;TCHARS,SCHARS1:WRD3;
    LINE:STRING[80];


PROCEDURE ENDOLN;
BEGIN
  COL:=1;IF LENGTH(P^.LINE) = 80 THEN COL :=80 ELSE
   BEGIN
    FOR I:= 1 TO LENGTH(P^.LINE) DO
     BEGIN
     COL:=COL+1;
     END;
   END;
END;

PROCEDURE INSERT_OVERWRITE;
BEGIN
  IF F THEN F:=FALSE
  ELSE F:=TRUE;
END;

PROCEDURE ENTERALINE;FORWARD;

PROCEDURE ENTERLINE;FORWARD;
```

```
PROCEDURE INCHARS;
LABEL XX1,PP,LL,XX,YY,ZZ,MM,VV,YY1;
VAR J,K,I,LIM,POS,inc,TCOL,T:INTEGER;
    WRAPWRD:STRING[40];
BEGIN
  IF F THEN
   BEGIN
        m:=0;
         FOR I := 1 to LENGTH(P^.LINE)  DO
          BEGIN
           IF P^.LINE[I] = ' ' THEN
              BEGIN
                M:=M+1;BLARR[M]:=I;
              END;
           END;
MM:    IF LENGTH(P^.LINE+C)+1 > 80 THEN
       BEGIN
          IF (M = 0) OR (M=1) THEN
            IF LENGTH(P^.LINE+C) <= 80 THEN GOTO PP ELSE
             BEGIN
                 WRAPWRD:='';GOTO YY1;
             END;
            WRAPWRD:=COPY(P^.LINE,BLARR[M]+1,
                          LENGTH(P^.LINE)-BLARR[M]+1);
            LIM:=80-BLARR[M]+1; J:=0;K:=1;
            FOR I:= 1 TO LENGTH(P^.LINE) DO
             BEGIN
              IF P^.LINE[I] = ' ' THEN K:=K+1 ELSE GOTO XX1;
             END;
XX1:             T:=K;
XX:          FOR I := K TO LENGTH(P^.LINE) DO
             BEGIN
                IF P^.LINE[I] = ' ' THEN
                 BEGIN
                   K:=I+2;J:=J+1;
                   IF J > LIM THEN GOTO YY;
                   INSERT(' ',P^.LINE,I);GOTO ZZ;
                 END;
             END;
             IF    (I = LENGTH(P^.LINE)) OR (I+1= 80) THEN
                   BEGIN K:=T;j:=j-1;GOTO VV; END;
ZZ:          GOTO XX;
VV:          IF J <= LIM THEN GOTO XX1;
  YY:        if p^.line[80] = ' ' then begin k:=t;j:=j-2;goto xx; en ;
  YY1:       GOTOXY(1,ROW);M:=0;
             WRITE(P^.LINE);
             IF P = RHEAD THEN ENTERALINE
               ELSE BEGIN ENTERLINE;GOTOXY(1,ROW);GOTO LL;END ;
             ROW:=ROW+1;GOTOXY(1,ROW);
LL:          P^.LINE:=WRAPWRD+C;WRITE(P^.LINE);ENDOLN;
        END
       ELSE
```

```
        BEGIN
PP:       INSERT(C,P^.LINE,COL);
          GOTOXY(1,ROW);
          WRITE(P^.LINE);
          IF COL = 80 THEN COL :=80 ELSE COL:=COL+1;
          GOTOXY(COL,ROW);
        END;
    END
   ELSE
     BEGIN
       A:=COPY(P^.LINE,1,COL-1);
       B:=COPY(P^.LINE,COL+1,80-COL);
       P^.LINE:=CONCAT(A,B);
       INSERT(C,P^.LINE,COL);
       GOTOXY(1,ROW);
       WRITE(P^.LINE);
       COL:=COL+1;
       GOTOXY(COL,ROW);
     END;
END;

PROCEDURE FORWRD;
BEGIN
  X:=P;P:=P^.RLINK;
END;

PROCEDURE BACKWARD;
BEGIN
 P:=X;X:=X^.LLINK;
END;

PROCEDURE LOADTEXTINLL;
BEGIN
  CLRSCR;
  WRITE(' NAME OF WORKFILE : ');
  READLN(TXTFN);
  ASSIGN(TXTFV,TXTFN);
  {$I-}RESET(TXTFV);{$I+}
  IF IORESULT <> 0 THEN
   BEGIN
    WRITELN(' NEW FILE : ',TXTFN);
    HIGHVIDEO;DELAY(2000);
    NEW(P);
    LHEAD:=P;P^.LLINK:=NIL;P^.RLINK:=NIL;
    RHEAD:=P;
    P^.LINE:='';
    CLOSE(TXTFV);
    EXIT;
   END
ELSE
 BEGIN
  WRITELN(' LOADING FILE : ',TXTFN);DELAY(1000);
  NEW(P);
  LHEAD:=P;P^.LLINK := NIL;
```

```
  WHILE NOT EOF(TXTFV) DO
    BEGIN
      P^.LINE := '';
      READLN(TXTFV,P^.LINE);
      NEW(Q);
      R:=P;P^.RLINK:=Q;Q^.LLINK:=P;P:=Q;
    END;
  IF P<> LHEAD THEN
    BEGIN
      R^.RLINK:=NIL;RHEAD:=R;DISPOSE(Q);
    END
  ELSE
    BEGIN
      DISPOSE(P);LHEAD:=NIL;RHEAD:=NIL;
    END;
 END;
  CLOSE(TXTFV);
END;

PROCEDURE SAVE_TEXT;
VAR P:LPTR;
BEGIN
{ GOTOXY(1,24);WRITELN('SAVING FILE : ',TXTFN);}
 ASSIGN(TXTFV,TXTFN);
 REWRITE(TXTFV);
 P:=LHEAD;
 WHILE P<>NIL DO
  BEGIN
   WRITELN(TXTFV,P^.LINE);
   P:=P^.RLINK;
  END;
  CLOSE(TXTFV);
END;

PROCEDURE PRINT_TEXT;
VAR P:LPTR;
BEGIN
 P:=LHEAD;
 WHILE P<>NIL DO
  BEGIN
   WRITELN(LST,P^.LINE);
   P:=P^.RLINK;
  END;
END;

PROCEDURE PRINT_FILE;
LABEL QQ;
VAR P:LPTR;
BEGIN
QQ:CLRSCR;
  WRITE(' NAME OF WORKFILE : ');
  READLN(TXTFN);
  ASSIGN(TXTFV,TXTFN);
  {$I-}RESET(TXTFV);{$I+}
```

```
  IF IORESULT <> 0 THEN
   BEGIN
    WRITELN(' FILE DOES NOT EXIST : ');
    GOTO QQ;
   END
ELSE
 BEGIN
  WRITELN(' LOADING FILE : ',TXTFN);DELAY(1000);
  NEW(P);
  LHEAD:=P;P^.LLINK := NIL;
  WHILE NOT EOF(TXTFV) DO
    BEGIN
      P^.LINE := '';
      READLN(TXTFV,P^.LINE);
      NEW(Q);
      R:=P;P^.RLINK:=Q;Q^.LLINK:=P;P:=Q;
    END;
   IF P<> LHEAD THEN
    BEGIN
      R^.RLINK:=NIL;RHEAD:=R;DISPOSE(Q);
    END
   ELSE
    BEGIN
      DISPOSE(P);LHEAD:=NIL;RHEAD:=NIL;
    END;
 END;
  CLOSE(TXTFV);
  WRITELN('PRINTING FILE ......................');
 P:=LHEAD;
 WHILE P<>NIL DO
  BEGIN
   WRITELN(LST,P^.LINE);
   P:=P^.RLINK;
  END;
END;

PROCEDURE DISPLAY1STLINE;
BEGIN
  WINDOW(1,1,80,3);LOWVIDEO; GOTOXY(1,1);
  IF FIRST THEN BEGIN CLREOL;GOTOXY(1,2);CLREOL;END;
  FIRST:=FALSE;
  GOTOXY(1,1);WRITE('COL : ',COL:2);
  GOTOXY(15,1);WRITE('ROW : ',ROW:2);
  GOTOXY(30,1);WRITE('FILE : ',TXTFN);
  GOTOXY(66,1);IF F THEN WRITE('INSERT   ':9)
          ELSE WRITE('OVERWRITE':9);
  GOTOXY(1,2);WRITE('F1 SAVE');GOTOXY(15,2);WRITE('F2 MAIN MENU');
  GOTOXY(30,2);WRITE('F3 DICT UPDATE');
  GOTOXY(50,2);WRITE('F4 PRINT ');
  GOTOXY(66,2);
  IF EDTR THEN WRITE('F10 PROMPT MODE':11);
  IF PRMT THEN WRITE('F10 EDIT MODE  ':11);
  WINDOW(1,3,80,25);HIGHVIDEO;
  IF EDTR THEN GOTOXY(COL,ROW) ELSE GOTOXY(1,1);
END;
```

```
PROCEDURE CURSORUP;
BEGIN
 IF P <> LHEAD THEN
   BEGIN
     IF ROW = 1 THEN
      BEGIN
        INSLINE;BACKWARD;
        GOTOXY(1,1);WRITE(P^.LINE);GOTOXY(COL,ROW);
      END
     ELSE
      BEGIN
        ROW := ROW-1;
        GOTOXY(COL,ROW);BACKWARD;
      END;
   END;
END;

PROCEDURE CURSORDOWN;
BEGIN
 IF P <> RHEAD THEN
   BEGIN
    IF ROW = 23 THEN
     BEGIN
       GOTOXY(1,1);
       DELLINE;GOTOXY(1,ROW);
       FORWRD;WRITE(P^.LINE);GOTOXY(COL,ROW);
     END
    ELSE
      BEGIN
         ROW:=ROW+1;FORWRD;
      END;
   END;
END;

PROCEDURE CURSORLEFT;
BEGIN
   IF COL = 1 THEN
    BEGIN
     COL := 80; CURSORUP;
    END
   ELSE
     COL := COL -1;
END;

PROCEDURE CURSORRIGHT;
BEGIN
   IF COL = 80 THEN
    BEGIN
     COL := 1; CURSORDOWN;
    END
   ELSE
     COL := COL +1;
END;
```

```
PROCEDURE PGDOWN;
LABEL A1;
BEGIN
  CLRSCR;
  IF (LHEAD = NIL) AND (RHEAD = NIL) THEN GOTO A1;
  FOR ROW := 1 TO 23 DO
    BEGIN
      GOTOXY(1,ROW);WRITE(P^.LINE);
      IF P <> RHEAD THEN FORWRD ELSE GOTO A1;
    END;
A1: COL := 1;
END;

PROCEDURE PGDOWN1;
LABEL A1;
BEGIN
  IF (LHEAD = NIL) AND (RHEAD = NIL) THEN GOTO A1;
  FOR ROW := 1 TO 23 DO
    BEGIN
      IF P <> RHEAD THEN FORWRD ELSE GOTO A1;
    END;
A1: COL := 1;
END;

PROCEDURE PGUP;
LABEL A2;
BEGIN
  FOR I := 1 TO 46 DO
    BEGIN
      IF P <> LHEAD THEN BACKWARD ELSE GOTO A2;
    END;
A2: PGDOWN;
END;

PROCEDURE PGUP1;
LABEL A2;
BEGIN
  FOR I := 1 TO 46 DO
    BEGIN
      IF P <> LHEAD THEN BACKWARD ELSE GOTO A2;
    END;
A2: PGDOWN1;
END;

PROCEDURE ENDOFFILE;
BEGIN
 WHILE P <> RHEAD DO
  FORWRD;
  PGUP1;PGDOWN;
  GOTOXY(1,1);DELLINE;
  GOTOXY(1,ROW);WRITE(RHEAD^.LINE);
  GOTOXY(COL,ROW);
END;
```

```
PROCEDURE BEGINOFFILE;
BEGIN
 WHILE P <> LHEAD DO
  BACKWARD;
 PGDOWN;
END;

PROCEDURE CNTRLHOME;
BEGIN
  FOR I:= ROW DOWNTO 1 DO
  IF P <> LHEAD THEN BACKWARD;
  ROW:=1;COL:=1;
END;

PROCEDURE CNTRLEND;
LABEL 30;
BEGIN
    FOR I:= ROW TO 23 DO
    IF P <> RHEAD THEN FORWRD ELSE GOTO 30;
30: IF P = RHEAD THEN
    BEGIN
     ROW :=I;COL:=1;
    END
  ELSE
    BEGIN
     ROW:=23;COL:=1;
    END;
    ENDOLN;
END;

PROCEDURE DELETEACHAR;
BEGIN
  A:=COPY(P^.LINE,1,COL-1);
  B:=COPY(P^.LINE,COL+1,80-COL);
  P^.LINE:=CONCAT(A,B);
  GOTOXY(1,ROW);
  CLREOL;
  WRITE(P^.LINE);GOTOXY(COL,ROW);
END;

PROCEDURE DELETEBACKCHAR;
BEGIN
  IF COL = 1 THEN EXIT;
  A:=COPY(P^.LINE,1,COL-2);
  B:=COPY(P^.LINE,COL,80-COL);
  P^.LINE:=CONCAT(A,B);
  GOTOXY(1,ROW);
  CLREOL;
  WRITE(P^.LINE);COL:=COL-1;GOTOXY(COL,ROW);
END;
```

```
PROCEDURE INSERTALINE;
BEGIN
 IF (LHEAD = NIL) AND (RHEAD=NIL) THEN
   BEGIN
      NEW(P);
      P^.LLINK:=NIL;P^.RLINK:=NIL;LHEAD:=P;RHEAD:=P;
      P^.LINE:='';
      WHILE KEYPRESSED DO
        BEGIN
          READ(KBD,C);WRITE(C);P^.LINE:=CONCAT(P^.LINE,C);
        END;
   END
 ELSE
   BEGIN
      INSLINE;
      IF (LHEAD = RHEAD) OR (P=LHEAD) THEN
        BEGIN
          NEW(Q);
          Q^.LLINK:=NIL;LHEAD:=Q;Q^.RLINK:=P;
          P^.LLINK:=Q;P:=Q;
          X:=NIL;
          P^.LINE:='';
          WHILE KEYPRESSED DO
            BEGIN
              READ(KBD,C);WRITE(C);P^.LINE:=CONCAT(P^.LINE,C);
            END;
        END
      ELSE BEGIN
           NEW(Z);
           X^.RLINK:=Z;Z^.LLINK:=X;
           P^.LLINK:=Z;Z^.RLINK:=P; P:=Z;
           P^.LINE:='';
           WHILE KEYPRESSED DO
             BEGIN
              READ(KBD,C);WRITE(C);P^.LINE:=CONCAT(P^.LINE,C);
             END;
           END;
   END;
END;


PROCEDURE ENTERLINE;
BEGIN
  ENDOLN;
  IF P <> RHEAD THEN
   BEGIN
     CURSORDOWN;COL:=1;DISPLAY1STLINE;INSERTALINE;
   END
 ELSE
```

```
   BEGIN
     NEW(Q);
     Q^.RLINK:=NIL;RHEAD:=Q;
     Q^.LLINK :=P; P^.RLINK:=Q;
     P:=Q;X:=P^.LLINK ;
     ROW:=ROW+1;COL:=1;GOTOXY(COL,ROW);
     P^.LINE:='';
     WHILE KEYPRESSED DO
     BEGIN
      READ(KBD,C);WRITE(C);P^.LINE:=CONCAT(P^.LINE,C);
     END;
  END;
END;

PROCEDURE ENTERALINE;
BEGIN
     ENDOLN;
     NEW(Q);Q^.RLINK:=NIL;RHEAD:=Q;Q^.LLINK :=P;
     P^.RLINK:=Q;
     P:=Q;X:=P^.LLINK ;
     P^.LINE:='';
END;

PROCEDURE DELETEALINE;
LABEL F1,F2,F3;
BEGIN
  IF (LHEAD = NIL) AND (RHEAD = NIL) THEN GOTO F1;
  IF LHEAD = RHEAD THEN
     BEGIN
       DISPOSE(LHEAD);LHEAD:= NIL;RHEAD := NIL;
       DELLINE;GOTO F1;
     END
  ELSE
     BEGIN
       IF (P = LHEAD) THEN
         BEGIN
           P := P^.RLINK;LHEAD:=P;
           DISPOSE(P^.LLINK);P^.LLINK:=NIL;delline;
         END
       ELSE
         BEGIN
            IF P = RHEAD THEN
              BEGIN
                DISPOSE(P);
                P := X; RHEAD :=P;X:= X^.LLINK;
                P^.RLINK := NIL;DELLINE;ROW:=ROW-1;GOTO F1;
              END
            ELSE
              BEGIN
                X^.RLINK:=P^.RLINK;P^.RLINK^.LLINK:=X;
                DISPOSE(P);P:=X^.RLINK;DELLINE;
              END
         END
     END;
Z:= X;Y:=P;TROW:= ROW;
```

```
FOR ROW := TROW TO 23 DO
 BEGIN
   IF P = RHEAD THEN GOTO F2 ELSE FORWRD;
 END;
 F2: IF ROW <> 23 THEN GOTO F3;
     GOTOXY(1,ROW);
     WRITE(P^.LINE);GOTOXY(COL,ROW);
F3:  X:= Z;P:=Y;
     ROW := TROW;
     GOTOXY(COL,ROW);
F1:END;


PROCEDURE WRITE_ALL_RECS;FORWARD;


PROCEDURE CURSORFUNCS;
LABEL L4;
BEGIN
  WINDOW(1,3,80,25);
  READ(KBD,C);
  IF C = #9 THEN BEGIN P^.LINE := P^.LINE+'          ';
  ENDOLN;EXIT;END;
  IF C IN [' '..'~'] THEN
   BEGIN
    INCHARS; GOTO L4;
   END;
  IF (C = #27) AND KEYPRESSED THEN READ(KBD,C);
  CASE ORD(C) OF
     8:DELETEBACKCHAR;{BACKSPACE}
    59:SAVE_TEXT;
    61:WRITE_ALL_RECS;
    62:PRINT_TEXT;
    60:begin QUIT_EDIT:=TRUE;EXIT;END;
    68:BEGIN FLAG:=TRUE;EDTR:=FALSE;PRMT:=TRUE;DISPLAY1STLINE;EXIT;END;
    71:COL:=1;{HOME}
    72:CURSORUP;
    73:PGUP;
    75:CURSORLEFT;
    77:CURSORRIGHT;
    79:ENDOLN;
    80:CURSORDOWN;
    81:PGDOWN;
    13:ENTERLINE;
    14:INSERTALINE;
    25:DELETEALINE;
    82:INSERT_OVERWRITE;
    83:DELETEACHAR;  {DEL}
    117:CNTRLEND;
    118:ENDOFFILE; {^PG DN}
    119:CNTRLHOME;
    132:BEGINOFFILE; {^PG UP}
  END; {of case}
L4:END;
```

```
PROCEDURE EDITOR;
LABEL L10;
VAR C:CHAR;
BEGIN {MAIN PROGRAM}
 IF FLAG1 THEN BEGIN FLAG1:=FALSE;LOADTEXTINLL;END;
 X:=NIL;P:=LHEAD;Q:=RHEAD; QUIT_EDIT:=FALSE;
 F:=TRUE;M:=0;
 WINDOW(1,3,80,25);CLRSCR;
 PGDOWN;
 P:=LHEAD;X:=NIL;ROW:=1;COL:=1;
 FIRST:=TRUE;
 L10:DISPLAY1STLINE;
     CURSORFUNCS;
    IF QUIT_EDIT THEN
      BEGIN
       GOTOXY(1,23);CLREOL;WRITE('FILE SAVE (Y/N) ? ');READ(KBD,C);
        IF (C='Y') OR (C ='y') THEN SAVE_TEXT;EXIT;
       END;
    IF FLAG THEN BEGIN FLAG:=FALSE;EXIT;END;
    GOTO L10;
END;

PROCEDURE INITIALISE_ALL_DIC;
VAR I: CHAR;
BEGIN
   FOR I := 'A' TO 'Z' DO
     BEGIN
       SWDIC[I].COUNT := 0;
       SWDIC[I].NEXT := NIL;
       SWDIC1[I].COUNT := 0;
       SWDIC1[I].NEXT := NIL;
       DWDIC[I].COUNT := 0;
       DWDIC[I].NEXT := NIL;
       TWDIC[I].COUNT := 0;
       TWDIC[I].NEXT := NIL;
     END;
END;


PROCEDURE CREATE_SW_DIC;
LABEL 10,20;
VAR D: CHAR;
    FOUND:BOOLEAN;
          TMP:INTEGER;
BEGIN
      D:= SWRD[1] ;IF NOT(D IN['A'..'Z']) THEN EXIT;
      IF SWDIC[D].NEXT = NIL THEN
       BEGIN
        NEW(SP);
        SP^.LLINK:= NIL;SP^.WORD := SWRD; SP^.FREQ :=
        SP^.RLINK := NIL;SWDIC[D].NEXT := SP;
        SWDIC[D].COUNT:=1;EXIT;
       END;
           SP:= SWDIC[D].NEXT;
```

```
        FOUND := FALSE;
        WHILE (SP <> NIL) AND (NOT FOUND) DO
{2}      BEGIN
            IF SP^.WORD = SWRD THEN
    {3}       BEGIN
                 FOUND := TRUE;
                 SP^.FREQ :=SP^.FREQ +1;
                 TMP := SP^.FREQ;
                 SX := SP;
                 IF SX^.LLINK = NIL THEN EXIT;
                 WHILE (SP^.LLINK <> NIL)
                        AND (SP^.LLINK^.FREQ < TMP) DO
                   BEGIN  SP:= SP^.LLINK; END;
                 IF SP = SX THEN EXIT;
                  IF SP^.LLINK = NIL THEN
       {4}          BEGIN
                       IF SX^.RLINK = NIL THEN
             {5}         BEGIN
                              SX^.LLINK^.RLINK := NIL;
                              SX^.LLINK:= NIL; SX^.RLINK:=SP;
                              SP^.LLINK:=SX;SWDIC[D].NEXT:=SX;
             {5}         END
                       ELSE
             {6}         BEGIN
                           SX^.LLINK^.RLINK:=SX^.RLINK;
                           SX^.RLINK^.LLINK:=SX^.LLINK;
                           SX^.LLINK:= NIL; SX^.RLINK:=SP;
                           SP^.LLINK:=SX;SWDIC[D].NEXT:=SX;
              {6}        END
       {4}           END
                     ELSE
           {7}         BEGIN
                           IF SX^.RLINK = NIL THEN
                            BEGIN
                               SX^.LLINK^.RLINK:=NIL;
                               SP^.LLINK^.RLINK := SX;
                               SX^.RLINK:=SP;SX^.LLINK:=SP^.LLINK;
                               SP^.LLINK := SX;
                            END
                            ELSE
                               BEGIN
                                  SX^.LLINK^.RLINK:=SX^.RLINK;
                                  SX^.RLINK^.LLINK:=SX^.LLINK;
                                  SP^.LLINK^.RLINK:=SX;
                                  SX^.RLINK:=SP;SX^.LLINK:=SP^.LLINK;
                                  SP^.LLINK := SX;
                               END
             {7}         END
         {3}       END
                   ELSE
                      BEGIN
                       SQ:=SP;
                       SP:=SP^.RLINK;
                      END;
 {2}      END  ;
```

```
20:         IF (NOT FOUND) AND (SQ<>NIL) AND (SP = NIL) THEN
              BEGIN
                NEW(SX);SX^.RLINK:=NIL;SX^.LLINK:=SQ;SX^.WORD:=SWRD;
                SX^.FREQ:=1;SQ^.RLINK:=SX;
                SWDIC[D].COUNT:=SWDIC[D].COUNT+1;
              END;
    );

    )CEDURE CREATE_DW_DIC;
    ( D:CHAR;
      DONE:BOOLEAN;
    ;IN
        D:= DWRD[1] ;IF NOT(D IN['A'..'Z']) THEN EXIT;
        IF DWDIC[D].NEXT = NIL THEN
         BEGIN
          NEW(DP);
          DP^.LLINK:= NIL;DP^.WORD := DWRD;DP^.FREQ:=1;
          DP^.RLINK := NIL;DWDIC[D].NEXT := DP;
          DWDIC[D].COUNT:=1;EXIT;
         END;
             DP:= DWDIC[D].NEXT;
             DONE := FALSE;
             WHILE (DP <> NIL) AND (NOT DONE) DO
      {2}     BEGIN
                 IF DP^.WORD = DWRD THEN
                  BEGIN
                      DONE := TRUE;
                      DP^.FREQ:=DP^.FREQ+1;
                  END
                 ELSE
                    IF DWRD < DP^.WORD THEN
                     BEGIN
                       NEW(DX);
                       DX^.RLINK:=DP;DX^.WORD:=DWRD;DX^.FREQ:=1;
                       DX^.LLINK:=DP^.LLINK;DP^.LLINK:=DX;
                       IF DX^.LLINK =NIL THEN
                        DWDIC[D].NEXT :=DX
                       ELSE
                         DX^.LLINK^.RLINK:=DX;
                      DWDIC[D].COUNT:=DWDIC[D].COUNT+1;
                       DONE :=TRUE;
                     END
                    ELSE
                     BEGIN
                        DQ1:=DP;
                        DP:=DP^.RLINK;
                     END;
        {2}     END  ;
                IF (NOT DONE) AND  (DP = NIL) THEN
                BEGIN
                 NEW(DX);DX^.RLINK:=NIL;DX^.LLINK:=DQ1;
                 DX^.WORD:=DWRD;DX^.FREQ:=1;
                 DQ1^.RLINK:=DX;DWDIC[D].COUNT:=DWDIC[D].COUNT+1;
                END;
    );
```

```
PROCEDURE CREATE_SW1_DIC;
VAR D:CHAR;
    DONE:BOOLEAN;
BEGIN
      D:= SWRD[1] ;IF NOT(D IN['A'..'Z']) THEN EXIT;
      IF SWDIC1[D].NEXT = NIL THEN
       BEGIN
        NEW(SP);
        SP^.LLINK:= NIL;SP^.WORD := SWRD;SP^.FREQ:=1;
        SP^.RLINK := NIL;SWDIC1[D].NEXT := SP;
        SWDIC1[D].COUNT:=1;EXIT;
       END;
           SP:= SWDIC1[D].NEXT;
           DONE := FALSE;
           WHILE (SP <> NIL) AND (NOT DONE) DO
    {2}     BEGIN
              IF SP^.WORD = SWRD THEN
               BEGIN
                   DONE := TRUE;
                   SP^.FREQ:=SP^.FREQ+1;
               END
              ELSE
                 IF SWRD < SP^.WORD THEN
                  BEGIN
                    NEW(SX);
                    SX^.RLINK:=SP;SX^.WORD:=SWRD;SX^.FREQ:
                    SX^.LLINK:=SP^.LLINK;SP^.LLINK:=SX;
                    IF SX^.LLINK =NIL THEN
                     SWDIC1[D].NEXT :=SX
                    ELSE
                      SX^.LLINK^.RLINK:=SX;
                   SWDIC1[D].COUNT:=SWDIC1[D].COUNT+1;
                    DONE :=TRUE;
                  END
                 ELSE
                  BEGIN
                   SQ1:=SP;
                    SP:=SP^.RLINK;
                  END;
     {2}      END  ;
              IF (NOT DONE) AND  (SP = NIL) THEN
              BEGIN
               NEW(SX);SX^.RLINK:=NIL;SX^.LLINK:=SQ1;
               SX^.WORD:=SWRD;SX^.FREQ:=1;
               SQ1^.RLINK:=SX;SWDIC1[D].COUNT:=SWDIC1[D].COUI
              END;
END;
```

```
PROCEDURE CREATE_TW_DIC;
VAR D:CHAR;
    DONE:BOOLEAN;
BEGIN
      D:= TWRD[1] ;IF NOT(D IN['A'..'Z']) THEN EXIT;
      IF TWDIC[D].NEXT = NIL THEN
       BEGIN
        NEW(TP);
        TP^.LLINK:= NIL;TP^.WORD := TWRD;TP^.FREQ:=1;
        TP^.RLINK := NIL;TWDIC[D].NEXT := TP;
        TWDIC[D].COUNT:=1;EXIT;
       END;
           TP:= TWDIC[D].NEXT;
           DONE := FALSE;
           WHILE (TP <> NIL) AND (NOT DONE) DO
    {2}     BEGIN
              IF TP^.WORD = TWRD THEN
               BEGIN
                   DONE := TRUE;
                   TP^.FREQ:=TP^.FREQ+1;
               END
              ELSE
                 IF TWRD < TP^.WORD THEN
                  BEGIN
                    NEW(TX);
                    TX^.RLINK:=TP;TX^.WORD:=TWRD;TX^.FREQ:=1;
                    TX^.LLINK:=TP^.LLINK;TP^.LLINK:=TX;
                    IF TX^.LLINK =NIL THEN
                     TWDIC[D].NEXT :=TX
                    ELSE
                      TX^.LLINK^.RLINK:=TX;
                   TWDIC[D].COUNT:=TWDIC[D].COUNT+1;
                    DONE :=TRUE;
                  END
                 ELSE
                  BEGIN
                   TQ1:=TP;
                    TP:=TP^.RLINK;
                  END;
      {2}     END  ;
              IF (NOT DONE) AND  (TP = NIL) THEN
              BEGIN
               NEW(TX);TX^.RLINK:=NIL;TX^.LLINK:=TQ1;
               TX^.WORD:=TWRD;TX^.FREQ:=1;
               TQ1^.RLINK:=TX;TWDIC[D].COUNT:=TWDIC[D].COUNT+1;
              END;
END;
```

```
PROCEDURE CREATE_SW_DIC1;
VAR D:CHAR;
BEGIN
      D:= SWRD[1] ;IF NOT(D IN['A'..'Z']) THEN EXIT;
      IF SWDIC[D].NEXT = NIL THEN
       BEGIN
        NEW(SP);
        SP^.LLINK:= NIL;SP^.WORD := SWRD; SP^.FREQ := FREQ;
        SP^.RLINK := NIL;SWDIC[D].NEXT := SP;
        SWDIC[D].COUNT:=1;EXIT;
       END;
       SP:=SWDIC[D].NEXT;
       WHILE SP<> NIL DO
            BEGIN
                     SQ:=SP;
                     SP:=SP^.RLINK;
             END;
      IF (SQ<>NIL) AND (SP = NIL) THEN
            BEGIN
              NEW(SX);SX^.RLINK:=NIL;SX^.LLINK:=SQ;SX^.WORD:=SWRD;
              SX^.FREQ:=FREQ;SQ^.RLINK:=SX;
              SWDIC[D].COUNT:=SWDIC[D].COUNT+1;
            END;
       SWDIC[D].COUNT:=SWNO;
END;

PROCEDURE CREATE_SW1_DIC1;
VAR D:CHAR;
BEGIN
      D:= SWRD[1] ;IF NOT(D IN['A'..'Z']) THEN EXIT;
      IF SWDIC1[D].NEXT = NIL THEN
       BEGIN
        NEW(SP);
        SP^.LLINK:= NIL;SP^.WORD := SWRD; SP^.FREQ := FREQ;
        SP^.RLINK := NIL;SWDIC1[D].NEXT := SP;
        SWDIC1[D].COUNT:=1;EXIT;
       END;
       SP:=SWDIC1[D].NEXT;
       WHILE SP<> NIL DO
            BEGIN
                     SQ:=SP;
                     SP:=SP^.RLINK;
             END;
      IF (SQ<>NIL) AND (SP = NIL) THEN
            BEGIN
              NEW(SX);SX^.RLINK:=NIL;SX^.LLINK:=SQ;SX^.WORD:=SWRD;
              SX^.FREQ:=FREQ;SQ^.RLINK:=SX;
              SWDIC1[D].COUNT:=SWDIC1[D].COUNT+1;
            END;
       SWDIC1[D].COUNT:=SWNO;
END;
```

```
PROCEDURE CREATE_DW_DIC1;
VAR D:CHAR;
BEGIN
      D:= DWRD[1] ;IF NOT(D IN['A'..'Z']) THEN EXIT;
      IF DWDIC[D].NEXT = NIL THEN
       BEGIN
        NEW(DP);
        DP^.LLINK:= NIL;DP^.WORD := DWRD; DP^.FREQ := FREQ;
        DP^.RLINK := NIL;DWDIC[D].NEXT := DP;
        DWDIC[D].COUNT:=1;EXIT;
       END;
       DP:=DWDIC[D].NEXT;
       WHILE DP<> NIL DO
            BEGIN
                     DQ:=DP;
                     DP:=DP^.RLINK;
             END;
      IF (DQ<>NIL) AND (DP = NIL) THEN
            BEGIN
              NEW(DX);DX^.RLINK:=NIL;DX^.LLINK:=DQ;DX^.WORD:=DWRD;
              DX^.FREQ:=FREQ;DQ^.RLINK:=DX;
              DWDIC[D].COUNT:=DWDIC[D].COUNT+1;
            END;
       DWDIC[D].COUNT:=DWNO;
END;

PROCEDURE CREATE_TW_DIC1;
VAR D:CHAR;
BEGIN
      D:= TWRD[1] ;IF NOT(D IN['A'..'Z']) THEN EXIT;
      IF TWDIC[D].NEXT = NIL THEN
       BEGIN
        NEW(TP);
        TP^.LLINK:= NIL;TP^.WORD := TWRD; TP^.FREQ := FREQ;
        TP^.RLINK := NIL;TWDIC[D].NEXT := TP;TWDIC[D].COUNT:=1;
        EXIT;
       END;
       TP:=TWDIC[D].NEXT;
       WHILE TP<> NIL DO
            BEGIN
                     TQ:=TP;
                     TP:=TP^.RLINK;
             END;
      IF (TQ<>NIL) AND (TP = NIL) THEN
            BEGIN
              NEW(TX);TX^.RLINK:=NIL;TX^.LLINK:=TQ;TX^.WORD:=TWRD;
              TX^.FREQ:=FREQ;TQ^.RLINK:=TX;
              TWDIC[D].COUNT:=TWDIC[D].COUNT+1;
            END;
       TWDIC[D].COUNT:=TWNO;
END;
```

```
PROCEDURE READ_SW_RECS;
BEGIN
   ASSIGN(SFV,'SWDIC.DIC');
   RESET(SFV);
   WHILE NOT EOF(SFV) DO
     BEGIN
       READ(SFV,SWR);
       SWNO :=  SWR.NOREC;
       FOR I := 1 TO SWNO DO
        BEGIN
         READ(SFV,SWR);
         SWRD:=SWR.WORD;FREQ:=SWR.FREQ;
         CREATE_SW_DIC1;
        END;
      END;
      CLOSE(SFV);
END;

PROCEDURE READ_SW1_RECS;
BEGIN
   ASSIGN(S1FV,'SWDIC1.DIC');
   RESET(S1FV);
   WHILE NOT EOF(S1FV) DO
     BEGIN
       READ(S1FV,SWR1);
       SWNO :=  SWR1.NOREC;
       FOR I := 1 TO SWNO DO
        BEGIN
         READ(S1FV,SWR1);
         SWRD:=SWR1.WORD;FREQ:=SWR1.FREQ;
         CREATE_SW1_DIC1;
        END;
      END;
      CLOSE(S1FV);
END;

PROCEDURE READ_DW_RECS;
BEGIN
   ASSIGN(DFV,'DWDIC.DIC');
   RESET(DFV);
   WHILE NOT EOF(DFV) DO
     BEGIN
       READ(DFV,DWR);
     DWNO :=  DWR.NOREC;
       FOR I := 1 TO DWNO DO
        BEGIN
         READ(DFV,DWR);
        DWRD:=DWR.WORD;FREQ:=DWR.FREQ;
         CREATE_DW_DIC1;
        END;
      END;
      CLOSE(DFV);
END;
```

```
PROCEDURE READ_TW_RECS;
BEGIN
   ASSIGN(TFV,'TWDIC.DIC');
   RESET(TFV);
   WHILE NOT EOF(TFV) DO
     BEGIN
       READ(TFV,TWR);
     TWNO :=  TWR.NOREC;
       FOR I := 1 TO TWNO DO
        BEGIN
         READ(TFV,TWR);
        TWRD:=TWR.WORD;FREQ:=TWR.FREQ;
         CREATE_TW_DIC1;
        END;
      END;
      CLOSE(TFV);
END;

PROCEDURE WRITE_SW_RECS;
VAR CH:CHAR;
BEGIN
   ASSIGN(SFV,'SWDIC.DIC');
   REWRITE(SFV);
   FOR CH := 'A' TO 'Z' DO
    BEGIN
     SWR.RECKIND:=RCNT;
     SWR.NOREC:=SWDIC[CH].COUNT;
     WRITE(SFV,SWR);
     SP:=SWDIC[CH].NEXT;
     IF SP<>NIL THEN
      BEGIN
        FOR J:= 1 TO SWDIC[CH].COUNT DO
        BEGIN
           SWR.WORD:=SP^.WORD;
           SWR.FREQ:=SP^.FREQ;
           SP:=SP^.RLINK;
           WRITE(SFV,SWR);
         END;
       END;
     END;
    CLOSE(SFV);
END;

PROCEDURE WRITE_SW1_RECS;
VAR CH:CHAR;
BEGIN
   ASSIGN(S1FV,'SWDIC1.DIC');
   REWRITE(S1FV);
   FOR CH := 'A' TO 'Z' DO
    BEGIN
     SWR1.RECKIND:=RCNT;
     SWR1.NOREC:=SWDIC1[CH].COUNT;
     WRITE(S1FV,SWR1);
     SP:=SWDIC1[CH].NEXT;
```

```
      IF SP<>NIL THEN
       BEGIN
         FOR J:= 1 TO SWDIC1[CH].COUNT DO
         BEGIN
            SWR1.WORD:=SP^.WORD;
            SWR1.FREQ:=SP^.FREQ;
            SP:=SP^.RLINK;
            WRITE(S1FV,SWR1);
          END;
        END;
       END;
     CLOSE(S1FV);
END;

PROCEDURE WRITE_DW_RECS;
VAR CH:CHAR;
BEGIN
   ASSIGN(DFV,'DWDIC.DIC');
   REWRITE(DFV);
   FOR CH := 'A' TO 'Z' DO
    BEGIN
     DWR.RECKIND:=RCNT;
     DWR.NOREC:=DWDIC[CH].COUNT;
     WRITE(DFV,DWR);
     DP:=DWDIC[CH].NEXT;
     IF DP<>NIL THEN
      BEGIN
         FOR J:= 1 TO DWDIC[CH].COUNT DO
         BEGIN
            DWR.WORD:=DP^.WORD;
            DWR.FREQ:=DP^.FREQ;
            DP:=DP^.RLINK;
            WRITE(DFV,DWR);
          END;
        END;
       END;
     CLOSE(DFV);
END;

PROCEDURE WRITE_TW_RECS;
VAR CH:CHAR;
BEGIN
   ASSIGN(TFV,'TWDIC.DIC');
   REWRITE(TFV);
   FOR CH := 'A' TO 'Z' DO
    BEGIN
     TWR.RECKIND:=RCNT;
     TWR.NOREC:='TWDIC[CH].COUNT;
     WRITE(TFV,TWR);
     TP:=TWDIC[CH].NEXT;
     IF TP<>NIL THEN
      BEGIN
         FOR J:= 1 TO TWDIC[CH].COUNT DO
         BEGIN
            TWR.WORD:=TP^.WORD;
```

```
            TWR.FREQ:=TP^.FREQ;
            TP:=TP^.RLINK;
            WRITE(TFV,TWR);
          END;
        END;
      END;
    CLOSE(TFV);
END;

PROCEDURE CREATE_SW_LIST;
BEGIN
 IF (LHD = NIL)  THEN
 BEGIN
  NEW(P2);
  P2^.RIGHT := NIL;
  P2^.PRD := PERIOD;
  P2^.DATA := SWRD;
  LHD := P2;
 END
 ELSE
 BEGIN
  NEW(Q2);
  Q2^.RIGHT :=NIL;
  P2^.RIGHT := Q2;
  Q2^.PRD := PERIOD;
  Q2^.DATA := SWRD;
  P2 := Q2;
 END;
END;



PROCEDURE WRITE_ALL_RECS;
BEGIN
 WRITE_SW_RECS;
 WRITE_SW1_RECS;
END;


PROCEDURE READTEXT;
LABEL 3;
BEGIN
{$I-}
 I:= 1;
 SWRD:='';
WHILE NOT SEEKEOF(FV) DO
BEGIN
3:READ(FV,C);
 C:=UPCASE(C);
```

```
WHILE
(NOT
(C IN ['-','%','(',')',' ','.',',',';','?','!',':',
     CHR(10),CHR(13),CHR(26)]))
    DO
     BEGIN
      SWRD := CONCAT(SWRD,C);
      READ(FV,C);C:=UPCASE(C);
     END;
   IF (SWRD = '')  THEN GOTO 3;
   IF (C=':') OR (C = '.') OR( C = '?') OR (C='!') OR (C=CHR(26))
                          THEN PERIOD :=TRUE ELSE PERIOD :=FALSE;
   CREATE_SW_LIST; CREATE_SW_DIC;
   CREATE_SW1_DIC; I := I+1;  SWRD:='';
END;
  {$I+}
END;

PROCEDURE CREATE_DW;
LABEL 60;
BEGIN
 I:=1;
 IF (LHD = NIL) THEN EXIT;
 Q2 := LHD;
60: P2 := Q2^.RIGHT;
IF P2 = NIL THEN EXIT;
IF ((NOT Q2^.PRD) AND (NOT P2^.PRD)) OR ((NOT Q2^.PRD)
     AND (P2^.PRD)) THEN
  BEGIN
         DWRD := CONCAT(Q2^.DATA,' ',P2^.DATA);
         CREATE_DW_DIC;
  END;
 Q2 :=P2;
 GOTO 60;
END;

PROCEDURE CREATE_TW;
LABEL 50;
BEGIN
 J:=1;
 IF (LHD = NIL) THEN EXIT;
 Q2 := LHD;
50: P2 := Q2^.RIGHT;
 IF P2 = NIL THEN EXIT;
 R2 :=P2^.RIGHT;
 IF R2 = NIL THEN EXIT;
 IF ((NOT Q2^.PRD) AND (NOT P2^.PRD) AND(NOT R2^.PRD)) OR
            ((NOT Q2^.PRD) AND (NOT P2^.PRD) AND(R2^.PRD)) THEN
  BEGIN
        TWRD := CONCAT(Q2^.DATA,' ',P2^.DATA,' ',R2^.DATA);
        CREATE_TW_DIC;
  END;
   Q2 :=P2;P2:=R2;
 GOTO 50;
END;
```

```
PROCEDURE PRINT_SW_DIC;
VAR D:CHAR;
  BEGIN
      INITIALISE_ALL_DIC;READ_SW_RECS;I:=0;
      FOR D:= 'A' TO 'Z' DO
       BEGIN
        SP:= SWDIC[D].NEXT;
        WHILE SP <> NIL DO
          BEGIN
             WRITELN(I+1:3,' ',SP^.FREQ:3,' ',SP^.WORD);
             DELAY(100);I:=I+1;
             SP:=SP^.RLINK;
          END;
      END;
      WRITELN;WRITELN('NO OF SINGLE WORDS IN THE DICTIONARY = ',I);
      DELAY(2000);
 END;

PROCEDURE PRINT_SW1_DIC;
VAR D:CHAR;
  BEGIN
      INITIALISE_ALL_DIC;READ_SW1_RECS;I:=0;
      FOR D:= 'A' TO 'Z' DO
       BEGIN
        SP:= SWDIC1[D].NEXT;
        WHILE SP <> NIL DO
          BEGIN
             WRITELN(I+1:3,' ',SP^.FREQ:3,' ',SP^.WORD);
             DELAY(100);I:=I+1;
             SP:=SP^.RLINK;
          END;
      END;
      WRITELN;WRITELN('NO OF SINGLE WORDS IN THE DICTIONARY = ',I);
      DELAY(2000);
 END;

PROCEDURE PRINT_DW_DIC;
VAR D:CHAR;
  BEGIN
      INITIALISE_ALL_DIC;READ_DW_RECS;I:=0;
      FOR D:= 'A' TO 'Z' DO
       BEGIN
        DP:= DWDIC[D].NEXT;
        WHILE DP <> NIL DO
          BEGIN
             WRITELN(I+1:3,' ',DP^.FREQ:3,' ',DP^.WORD);
             DELAY(100);I:=I+1;
             DP:=DP^.RLINK;
          END;
      END;
      WRITELN;WRITELN('NO OF DOUBLE WORDS IN THE DICTIONARY = ',I);
      DELAY(2000);
 END;
```

```
PROCEDURE PRINT_TW_DIC;
VAR D:CHAR;
  BEGIN
       INITIALISE_ALL_DIC;READ_TW_RECS;I:=0;
       FOR D:= 'A' TO 'Z' DO
        BEGIN
         TP:=TWDIC[D].NEXT;
         WHILE TP <> NIL DO
           BEGIN
              WRITELN(I+1:3,' ',TP^.FREQ:3,' ',TP^.WORD);
              DELAY(100);I:=I+1;
              TP:=TP^.RLINK;
           END;
       END;
       WRITELN;WRITELN('NO OF TRIPLE WORDS IN THE DICTIONARY = ',I);
       DELAY(2000);
 END;

PROCEDURE CREATE_SW_ARRAY_P; {PASS CHAR D}
VAR LIM:INTEGER;
BEGIN
      SWNO:=SWDIC[D].COUNT;
      SP:=SWDIC[D].NEXT;
      FOR I:= 1 TO SWNO DO
       BEGIN
            SWARR[I].WORD:=SP^.WORD;
            SWARR[I].FREQ:=SP^.FREQ;
            SP:=SP^.RLINK;{WRITELN(SWARR[I].WORD);}
       END;WINDOW(1,5,30,11);CLRSCR;
       IF SWNO < 5 THEN LIM := SWNO ELSE LIM := 5;
       FOR I := 1 TO LIM DO
        WRITELN(I:3,' ',SWARR[I].WORD);
        WINDOW(1,3,80,25);
END;

PROCEDURE CREATE_SW_ARRAY; {PASS CHAR D}
BEGIN
      SWNO:=SWDIC[D].COUNT;
      SP:=SWDIC[D].NEXT;
      FOR I:= 1 TO SWNO DO
       BEGIN
            SWARR[I].WORD:=SP^.WORD; SWARR[I].FREQ:=SP^.FREQ;
            SP:=SP^.RLINK;{WRITELN(SWARR[I].WORD);}
       END;
END;

PROCEDURE CREATE_SW1_ARRAY; {PASS CHAR D}
BEGIN
      SWNO:=SWDIC1[D].COUNT;SP:=SWDIC1[D].NEXT;
      FOR I:= 1 TO SWNO DO
       BEGIN
            SWARR1[I].WORD:=SP^.WORD; SWARR1[I].FREQ:=SP^.FREQ;
            SP:=SP^.RLINK;
       END;
END;
```

```
PROCEDURE CREATE_DW_ARRAY; {PASS CHAR D}
BEGIN
     DWNO:=DWDIC[D].COUNT;
     DP:=DWDIC[D].NEXT;
     FOR I:= 1 TO DWNO DO
      BEGIN
           DWARR[I].WORD:=DP^.WORD;
           DWARR[I].FREQ:=DP^.FREQ;
           DP:=DP^.RLINK;
      END;
END;

PROCEDURE CREATE_TW_ARRAY; {PASS CHAR D}
BEGIN
     TWNO:=TWDIC[D].COUNT;
     TP:=TWDIC[D].NEXT;
     FOR I:= 1 TO TWNO DO
      BEGIN
           TWARR[I].WORD:=TP^.WORD;
           TWARR[I].FREQ:=TP^.FREQ;
           TP:=TP^.RLINK;
      END;
END;

PROCEDURE SW_LINEAR_SRCH; {PASS SCHARS}
LABEL X,Y;
VAR FPOS,LPOS,I,J,LIMIT,LIM:INTEGER;
    TSCHARS : WRD1;
BEGIN
  FOUND:=FALSE;
  FPOS :=0;
  FOR I:= 1 TO SWNO DO
   BEGIN
     TSCHARS :=COPY(SWARR1[I].WORD,1,LENGTH(SCHARS));
     IF TSCHARS = SCHARS THEN
       BEGIN
        FPOS:=I;
        FOUND:=TRUE;
        GOTO X;
       END;
   END;
X:LPOS := FPOS;
  IF FOUND THEN
   BEGIN
        FOR I:= FPOS TO SWNO DO
          BEGIN
               TSCHARS :=COPY(SWARR1[I].WORD,1,LENGTH(SCHARS)
               IF TSCHARS = SCHARS THEN
                 LPOS := I
               ELSE GOTO Y;
          END;
    Y: FOR I:= FPOS TO LPOS DO
         BEGIN
              J:= (I-FPOS)+1;
```

```
                SWARR2[J]:=SWARR1[I];
            END;
          LIMIT := (LPOS-FPOS)+1;
          FOR I := 1 TO LIMIT-1 DO
           FOR J := I+1 TO LIMIT DO
             BEGIN
                  IF SWARR2[I].FREQ < SWARR2[J].FREQ THEN
                    BEGIN
                      TSR := SWARR2[I];
                      SWARR2[I]:=SWARR2[J];
                      SWARR2[J]:=TSR;
                    END;
             END;
             WINDOW(1,5,30,11);CLRSCR;
           IF LIMIT < 5 THEN LIM:=LIMIT ELSE LIM :=5;
           FOR I:= 1 TO LIM DO
              WRITELN(I:3,' ',SWARR2[I].WORD);
              WINDOW(1,3,80,25);
   END;
END;

PROCEDURE DW_LINEAR_SRCH; {PASS DCHARS}
LABEL X,Y,X1;
VAR FPOS,LPOS,I,J,LIMIT,LIM:INTEGER;
    TDCHARS,TYYY,TXXX : WRD2;
BEGIN
  FOUND:=FALSE; FPOS :=0;
  FOR I:= 1 TO DWNO DO
   BEGIN
     TDCHARS :=COPY(DWARR[I].WORD,1,LENGTH(DCHARS));
     IF TDCHARS = DCHARS THEN
       BEGIN
        FPOS:=I;
        FOUND:=TRUE;
        GOTO X;
       END;
   END;
X:LPOS := FPOS;
  IF FOUND THEN
   BEGIN
        FOR I:= FPOS TO DWNO DO
          BEGIN
               TDCHARS :=COPY(DWARR[I].WORD,1,LENGTH(DCHARS));
               IF TDCHARS = DCHARS THEN
                 LPOS := I
               ELSE GOTO Y;
          END;
    Y: FOR I:= FPOS TO LPOS DO
         BEGIN
              J:= (I-FPOS)+1;
              DWARR2[J]:=DWARR[I];
         END;
       LIMIT := (LPOS-FPOS)+1;
       FOR I := 1 TO LIMIT-1 DO
        FOR J := I+1 TO LIMIT DO
```

```
          BEGIN
               IF DWARR2[I].FREQ < DWARR2[J].FREQ THEN
                 BEGIN
                   TDR := DWARR2[I];
                   DWARR2[I]:=DWARR2[J];
                   DWARR2[J]:=TDR;
                 END;
          END;
          WINDOW(1,5,30,11);CLRSCR;
        IF LIMIT < 5 THEN LIM:=LIMIT ELSE LIM :=5;
        TYYY:='';
        FOR I:= 1 TO LIM DO
          BEGIN
           TXXX:=DWARR2[I].WORD;
           FOR J := 1 TO LENGTH(DWARR2[I].WORD) DO
              IF TXXX[J] = ' ' THEN GOTO X1 ;
        X1:FOR K:= J+1 TO LENGTH(TXXX) DO
             TYYY:=TYYY+TXXX[K];
           WRITELN(I:3,' ',TYYY);TYYY:='';TXXX:='';
           END;
           WINDOW(1,3,80,25);
   END;
END;

PROCEDURE TW_LINEAR_SRCH; {PASS TCHARS O/P FOUND}
LABEL X,Y,X1,X2;
VAR FPOS,LPOS,I,J,K,KK,LIMIT,LIM:INTEGER;
    TTCHARS,TXXX,TYYY: WRD3;
BEGIN
  FOUND:=FALSE; FPOS :=0;
  FOR I:= 1 TO TWNO DO
   BEGIN
     TTCHARS :=COPY(TWARR[I].WORD,1,LENGTH(TCHARS));
     IF TTCHARS = TCHARS THEN
       BEGIN
        FPOS:=I;
        FOUND:=TRUE;
        GOTO X;
       END;
   END;
X:LPOS := FPOS;
  IF FOUND THEN
   BEGIN
        FOR I:= FPOS TO TWNO DO
          BEGIN
               TTCHARS :=COPY(TWARR[I].WORD,1,LENGTH(TCHARS));
               IF TTCHARS = TCHARS THEN
                 LPOS := I
               ELSE GOTO Y;
          END;
    Y: FOR I:= FPOS TO LPOS DO
         BEGIN
              J:= (I-FPOS)+1;
              TWARR2[J]:=TWARR[I];
         END;
```

```
        LIMIT := (LPOS-FPOS)+1;
        FOR I := 1 TO LIMIT-1 DO
         FOR J := I+1 TO LIMIT DO
           BEGIN
                IF TWARR2[I].FREQ < TWARR2[J].FREQ THEN
                  BEGIN
                    TTR := TWARR2[I];
                    TWARR2[I]:=TWARR2[J];
                    TWARR2[J]:=TTR;
                  END;
           END;
           WINDOW(1,5,30,11);CLRSCR;
         IF LIMIT < 5 THEN LIM:=LIMIT ELSE LIM :=5;
         TYYY:='';
         FOR I:= 1 TO LIM DO
           BEGIN
            TXXX:=TWARR2[I].WORD;
            FOR J := 1 TO LENGTH(TWARR2[I].WORD) DO
               IF TXXX[J] = ' ' THEN GOTO X1 ;
         X1:FOR K:= J+1 TO LENGTH(TXXX) DO
                IF TXXX[K] = ' ' THEN GOTO X2 ;
         X2:FOR KK:= K+1 TO LENGTH(TXXX) DO
              TYYY:=TYYY+TXXX[KK];
            WRITELN(I:3,' ',TYYY);TYYY:='';TXXX:='';
            END;
            WINDOW(1,3,80,25);
   END;
END;


PROCEDURE CREATE_FST_DIC;
LABEL 10;
BEGIN
10:  WRITE('GIVE NAME OF FILE TO BE USED : ');READLN(FNAME);
    ASSIGN(FV,FNAME);
    {$I-}
    RESET(FV);
    {$I+}
    IF IORESULT <> 0 THEN GOTO 10;
    INITIALISE_ALL_DIC;
    READTEXT;
    WRITE_SW_RECS;WRITE_SW1_RECS;
    CREATE_DW;
    CREATE_TW;
    WRITE_DW_RECS;WRITE_TW_RECS;
    CLOSE(FV);
END;
```

```
PROCEDURE UPDATE_DIC;
LABEL 20,30,40;
VAR C:CHAR;
BEGIN
20:  WRITE('GIVE NAME OF FILE TO BE USED FOR UPDATION : ');
  READLN(FNAME2);ASSIGN(FV,FNAME2);
  {$I-} RESET(FV); {$I+}
  IF IORESULT <> 0 THEN GOTO 20;
  INITIALISE_ALL_DIC;WRITELN('UPDATING.....WAIT!');
  READ_SW_RECS;READ_SW1_RECS;READ_DW_RECS;READ_TW_RECS;
  READTEXT;CREATE_DW;CREATE_TW;
40:  WRITE('DO YOU WANT TO UPDATE USING ANOTHER FILE (Y/N) ? ');
  READ(KBD,C);
  IF (C = 'Y') OR (C = 'y') THEN
   BEGIN
30: WRITE('GIVE NAME OF FILE TO BE USED FOR UPDATION : ');
    READLN(FNAME2);ASSIGN(FV,FNAME2);
    {$I-} RESET(FV); {$I+}
    IF IORESULT <> 0 THEN GOTO 30;
    WRITELN('UPDATING.....WAIT!');
    READTEXT;CREATE_DW;CREATE_TW;
    GOTO 40;
   END;
  WRITE_SW_RECS;WRITE_SW1_RECS;WRITE_DW_RECS;WRITE_TW_RECS;
  CLOSE(FV);
END;

PROCEDURE CREATE_ALL_ARR_P;
BEGIN
 CREATE_SW_ARRAY_P;
 CREATE_SW1_ARRAY;
 CREATE_DW_ARRAY;
 CREATE_TW_ARRAY;
END;

PROCEDURE CREATE_ALL_ARR;
BEGIN
 CREATE_SW_ARRAY;
 CREATE_SW1_ARRAY;
 CREATE_DW_ARRAY;
 CREATE_TW_ARRAY;
END;

PROCEDURE READ_ALL_RECS;
BEGIN
 INITIALISE_ALL_DIC;
 READ_SW_RECS;
 READ_SW1_RECS;
 READ_DW_RECS;
 READ_TW_RECS;
END;
```

```
PROCEDURE PRINT_ALL_DICS;
LABEL XX;
BEGIN
XX:WINDOW(1,1,80,25);CLRSCR;
 GOTOXY(27,12);WRITELN('1: DISPLAY ALL DICTIONARIES    ');
 GOTOXY(27,13);WRITELN('2: DISPLAY SINGLE WORD DICT (SWDIC.DIC) ');
 GOTOXY(27,14);WRITELN('3: DISPLAY SINGLE WORD DICT (SWDIC1.DIC) ');
 GOTOXY(27,15);WRITELN('4: DISPLAY DOUBLE WORD DICT (DWDIC.DIC) ');
 GOTOXY(27,16);WRITELN('5: DISPLAY TRIPLE WORD DICT (TWDIC.DIC) ');
 GOTOXY(27,17);WRITELN('6: QUIT' );
    GOTOXY(20,20);
    WRITE('PLEASE ENTER YOUR CHOICE (1,2,3,4,5,6) : ');READ(KBD,CI);
    WRITELN(CI);
    CASE CI OF
      '1':BEGIN
                  PRINT_SW_DIC;
                  PRINT_SW1_DIC;
                  PRINT_DW_DIC;
                  PRINT_TW_DIC;
          END;
      '2':PRINT_SW_DIC;
      '3':PRINT_SW1_DIC;
      '4':PRINT_DW_DIC;
      '5':PRINT_TW_DIC;
      '6':EXIT;
     END;
GOTO XX;
END;



PROCEDURE STORE_IN_LINE;
LABEL LL,XX,YY,ZZ,XX1,VV;
VAR J,K,LIM,I,T:INTEGER;
BEGIN
 clrscr;
 IF P^.LINE ='' THEN
    BEGIN
      IF SCHARS1[1] = ' ' THEN DELETE(SCHARS1,1,1);
        P^.LINE := SCHARS1
    END
  ELSE
       BEGIN
         IF LENGTH(P^.LINE)+LENGTH(SCHARS1)+1 <= 80 THEN
         P^.LINE :=P^.LINE+SCHARS1 ELSE
          BEGIN
            K:=1;J:=0; LIM := 80-LENGTH(P^.LINE);
            FOR I := 1 TO LENGTH(P^.LINE) DO
             BEGIN
               IF P^.LINE[I] = ' ' THEN k:=k+1 ELSE GOTO XX1;
             END;
XX1:         T:=K;
XX:         FOR I := K TO LENGTH(P^.LINE) DO
             BEGIN
                IF P^.LINE[I] = ' ' THEN
```

```
                BEGIN
                  K:=I+2;J:=J+1;IF J > LIM THEN GOTO YY;
                  INSERT(' ',P^.LINE,I);GOTO ZZ;
                END;
             END;
             IF (I = LENGTH(P^.LINE)) OR (I+1=80) THEN
              BEGIN
               K:=T;J:=J-1;GOTO VV;
              END;
ZZ:         GOTO XX;
VV:         IF J <= LIM THEN GOTO XX1;
YY:         IF P^.LINE[80]  = ' ' THEN
               BEGIN
                  K:=T;J:=J-2;GOTO XX;
               END;
             ENTERALINE;
             P^.LINE:='';
             IF SCHARS1[1] = ' ' THEN DELETE(SCHARS1,1,1);
             P^.LINE:=SCHARS1;
           END;
        END;
   WRITELN(P^.LINE);
 END;

PROCEDURE EFFICIENCY;
BEGIN
       CLRSCR;
       WRITELN('ACTUAL KEY STROKES MADE UPTIL NOW = ',KEYCNT);
       WRITELN('NO OF CHARACTERS UPTIL NOW  = ',LEN);
       WRITELN('EFFICIENCY UPTIL NOW = ',
                (100-((KEYCNT/LEN)*100)):4:2);
       DELAY(1100);
END;

PROCEDURE EDIT_TEXT;
LABEL L,L1,L2,L3,L4,L6,L8,N1,N2,N3;
VAR SCH :CHAR;
    CL:INTEGER;
BEGIN
    WINDOW(1,3,80,25);
    IF FLAG2 THEN
      BEGIN
        FLAG2:=FALSE; READ_ALL_RECS;
      END;
    LINE:='';SCHARS1:='';P:=RHEAD;QUIT_EDIT:=FALSE;
L2:STORE_IN_LINE;CL:=0;
   WRITE('GIVE FIRST CHAR : ');READ(KBD,D);D:=UPCASE(D);
   IF D = #13 THEN
    BEGIN
     IF P = RHEAD THEN ENTERALINE ELSE ENTERLINE;P^.LINE:='';
     SCHARS1:='';LEN:=LEN+LENGTH(SCHARS1);GOTO L2;
    END;
   IF (D = #27 ) AND KEYPRESSED THEN
    BEGIN
      READ(KBD,D);
```

```
      IF D =#60 THEN BEGIN EFFICIENCY; QUIT_EDIT:=TRUE; EXIT; END;
      IF D =#68 THEN
         BEGIN EFFICIENCY;PRMT:=FALSE;EDTR:=TRUE;
         DISPLAY1STLINE;EXIT;END;
    END;
   D:=UPCASE(D);WRITELN(D);
   IF D = #9 THEN BEGIN SCHARS1 :='          ';GOTO L2;END;
   KEYCNT:=KEYCNT+1;
   IF D IN  ['!'..'@',' ','['..'`','{'..'~'] THEN
     BEGIN
      SCHARS1:=D;LEN:=LEN+LENGTH(SCHARS1);GOTO L2;
     END;
L8:CREATE_ALL_ARR_P;
   {WRITE(' SELECT ANY WORD OR TYPE NEXT CHAR: ')}
    CL:=CL+1;GOTOXY(CL,14); WRITE(D);
   READ(KBD,SCH); SCH:=UPCASE(SCH);
   IF SCH = #13 THEN
    BEGIN
     IF P = RHEAD THEN ENTERALINE ELSE ENTERLINE;P^.LINE:='';
     SCHARS1:='';LEN:=LEN+LENGTH(SCHARS1);GOTO L2;
    END;
      IF (SCH = #27 ) AND KEYPRESSED THEN
    BEGIN
      READ(KBD,SCH);
      IF SCH =#60 THEN BEGIN EFFICIENCY;QUIT_EDIT:=TRUE; EXIT; END;
      IF SCH =#68 THEN
      BEGIN EFFICIENCY;PRMT:=FALSE;EDTR:=TRUE;DISPLAY1STLINE;
      EXIT;END;
    END;
     IF NOT (SCH IN ['0'..'9']) THEN
                 BEGIN
                   CL:=CL+1;GOTOXY(CL,14);WRITE(SCH);
                  END;
   KEYCNT:=KEYCNT+1;
   IF SCH IN ['1'..'5'] THEN
      BEGIN
           INDEX := ORD(SCH)-48;
           TSWRD:=SWARR[INDEX].WORD;
           GOTOXY(1,14);CLREOL;WRITE(TSWRD);DELAY(400);
      END
   ELSE
      BEGIN
           IF SCH <> ' ' THEN SCHARS := D+SCH;
           IF SCH = ' ' THEN BEGIN SCHARS:=D;END;
L:         IF SCH IN  [' ',',','.',';','?','!'] THEN
             BEGIN
               SWRD:=SCHARS;{WRITELN('KKK',SCHARS);}
               CREATE_SW_DIC;CREATE_SW1_DIC;SCHARS1:=' '+SCHARS;
              { WRITELN(' I =',SCHARS1);DELAY(1000);}
               LEN:=LEN+LENGTH(SCHARS1);
               GOTO L2;
             END;
L1:        SW_LINEAR_SRCH;
           IF FOUND THEN
            BEGIN
```

```
    READ(KBD,SCH); SCH:=UPCASE(SCH);
    IF SCH = #13 THEN
       BEGIN
        IF P = RHEAD THEN ENTERALINE ELSE ENTERLINE;
        P^.LINE:='';
        SCHARS1:='';LEN:=LEN+LENGTH(SCHARS1);GOTO L2;
       END;
       IF (SCH = #27 ) AND KEYPRESSED THEN
         BEGIN
           READ(KBD,SCH);
            IF SCH =#60 THEN
              BEGIN EFFICIENCY;QUIT_EDIT:=TRUE; EXIT; END;
            IF SCH =#68 THEN
               BEGIN
                EFFICIENCY;PRMT:=FALSE;
                EDTR:=TRUE;DISPLAY1STLINE;EXIT;
               END
         END;
    IF NOT (SCH IN ['0'..'9']) THEN
       BEGIN
         CL:=CL+1;GOTOXY(CL,14);WRITE(SCH);
       END;
    KEYCNT:=KEYCNT+1;
    IF SCH IN ['1'..'5'] THEN
      BEGIN
           INDEX := ORD(SCH)-48;
           TSWRD:=SWARR2[INDEX].WORD;
           GOTOXY(1,14);CLREOL;WRITE(TSWRD);
            DELAY(400);
      END
    ELSE
      BEGIN
        IF SCH <> ' ' THEN SCHARS := SCHARS+SCH;
        IF SCH IN  [' ',',','.',';','?','!'] THEN
        BEGIN
          SWRD:=SCHARS;CREATE_SW_DIC;CREATE_SW1_DIC;
          SCHARS1:=' '+SCHARS;LEN:=LEN+LENGTH(SCHARS1);
          GOTO L2;
        END;
        GOTO L1;
      END;
 END
ELSE
   BEGIN
    READ(KBD,SCH); SCH:=UPCASE(SCH);
     IF NOT (SCH IN ['0'..'9']) THEN
      BEGIN
    CL:=CL+1;GOTOXY(CL,14);WRITE(SCH);
      END;
    IF SCH = #13 THEN
      BEGIN
        IF P = RHEAD THEN ENTERALINE ELSE ENTERLINE;
        P^.LINE:='';
        SCHARS1:='';LEN:=LEN+LENGTH(SCHARS1);GOTO L2;
      END;
```

```
                IF (SCH = #27 ) AND KEYPRESSED THEN
    BEGIN
      READ(KBD,SCH);
            IF SCH =#60 THEN BEGIN EFFICIENCY;QUIT_EDIT:=TRUE;
            EXIT; END;
      IF SCH =#68 THEN
       BEGIN EFFICIENCY;PRMT:=FALSE;EDTR:=TRUE;DISPLAY1STLINE;
       EXIT;END;
    END;
              IF SCH <> ' ' THEN SCHARS:=SCHARS+SCH;
              KEYCNT:=KEYCNT+1;GOTO L;
             END
        END;
   DCHARS := TSWRD+' ';SWRD3:='';DCHARST:=TSWRD;
L3:DW_LINEAR_SRCH;
   IF FOUND THEN
      BEGIN
        READ(KBD,SCH); SCH:=UPCASE(SCH);
         IF NOT (SCH IN ['0'..'9']) THEN
                BEGIN
         CL:=CL+1;GOTOXY(CL,14);WRITE(SCH);
                END;
           IF SCH = #13 THEN
                BEGIN
                  IF P = RHEAD THEN ENTERALINE ELSE ENTERLINE;
                  P^.LINE:='';
                  SCHARS1:='';LEN:=LEN+LENGTH(SCHARS1);GOTO L2;
                END;
          IF (SCH = #27 ) AND KEYPRESSED THEN
    BEGIN
      READ(KBD,SCH);
            IF SCH =#60 THEN BEGIN EFFICIENCY;QUIT_EDIT:=TRUE;
            EXIT; END;
      IF SCH =#68 THEN
       BEGIN EFFICIENCY;PRMT:=FALSE;EDTR:=TRUE;DISPLAY1STLINE;
       EXIT;END;
    END;
        KEYCNT:=KEYCNT+1;
        IF SCH IN ['1'..'5'] THEN
          BEGIN
               INDEX:=ORD(SCH)-48;
               DWRD:=DWARR2[INDEX].WORD;
               GOTOXY(1,14);CLREOL;WRITE(DWRD);DELAY(400);
               WHILE KEYPRESSED DO;
          END
        ELSE
          BEGIN
               IF SCH <> ' ' THEN DCHARS:=DCHARS+SCH;
               IF SCH IN  [' ',',','.',';','?','!'] THEN
                BEGIN
                  SCHARS1:=' '+DCHARS;LEN:=LEN+LENGTH(SCHARS1);
                  GOTO L2;
                END;
                SWRD3:=SWRD3+SCH;{WRITELN(' SWRD3 = ',SWRD3);}
               GOTO L3;
```

```
          END
       END
    ELSE
       BEGIN
            SCHARS1:=' '+DCHARST;LEN:=LEN+LENGTH(SCHARS1);
            STORE_IN_LINE;
            IF SWRD3 = '' THEN
               BEGIN SCHARS1:='';LEN:=LEN+LENGTH(SCHARS1);GOTO L2;END;
            IF LENGTH(SWRD3) = 1 THEN
              BEGIN
                   D:=SWRD3;IF NOT (D IN ['A'..'Z']) THEN
                    BEGIN
                     SCHARS1:=' '+D;LEN:=LEN+LENGTH(SCHARS1);GOTO L2;
                    END;
                   GOTO L8;
              END
            ELSE
               BEGIN
                 D:=SWRD3[1];
                 IF  NOT (D IN ['A'..'Z']) THEN
                    BEGIN
                     SCHARS1:=' '+SWRD3;LEN:=LEN+LENGTH(SCHARS1);
                     GOTO L2;
                    END;
                 SCHARS:=SWRD3;CREATE_ALL_ARR;GOTO L1;
               END;
       END;
N3:    SWRD2:='';SWRD1:='';
       FOR I:= 1 TO LENGTH(DWRD) DO
       IF DWRD[I] = ' ' THEN GOTO N1 ELSE SWRD1:=SWRD1+DWRD[I];
n1:      for j:= i+1 to length(dwrd) do
       if dwrd[j] = ' ' then goto n2 else swrd2:= swrd2+dwrd[j];
N2:     TCHARS:=DWRD+' ';SWRD3:='';
        DWRD:='';
L4: TW_LINEAR_SRCH;
    IF FOUND THEN
       BEGIN
            READ(KBD,SCH); SCH:=UPCASE(SCH);
             IF NOT (SCH IN ['0'..'9']) THEN
                 BEGIN
             CL:=CL+1;GOTOXY(CL,14);WRITE(SCH);
                 END;
               IF SCH = #13 THEN
                 BEGIN
                   IF P = RHEAD THEN ENTERALINE ELSE ENTERLINE;
                   P^.LINE:='';
                   SCHARS1:='';LEN:=LEN+LENGTH(SCHARS1);GOTO L2;
                 END;
   IF (SCH = #27 ) AND KEYPRESSED THEN
    BEGIN
      READ(KBD,SCH);
            IF SCH =#60 THEN BEGIN EFFICIENCY;QUIT_EDIT:=TRUE;
            EXIT; END;
      IF SCH =#68 THEN
        BEGIN EFFICIENCY;PRMT:=FALSE;EDTR:=TRUE;DISPLAY1STLINE;
```

```
        EXIT;END;
    END;
            KEYCNT:=KEYCNT+1;
            IF SCH IN ['1'..'5'] THEN
               BEGIN
                    INDEX:= ORD(SCH)-48;
                    TWRD:=TWARR2[INDEX].WORD;
                     GOTOXY(1,14);CLREOL;WRITE(TWRD);DELAY(400);
               END
            ELSE
               BEGIN
                    IF SCH <> ' ' THEN TCHARS:=TCHARS+SCH;


                    IF SCH IN  [' ',',','.',';','?','!'] THEN
                     BEGIN
                        SCHARS1:=' '+TCHARS;
                        LEN:=LEN+LENGTH(SCHARS1);
                        GOTO L2;
                     END;
                     SWRD3:=SWRD3+SCH;
                    GOTO L4;
               END
       END
    ELSE
       BEGIN
            SCHARS1:=' '+SWRD1;LEN:=LEN+LENGTH(SCHARS1);STORE_IN_LINE
            IF SWRD3 = '' THEN
                 DCHARS :=SWRD2+' ' ELSE DCHARS := SWRD2+' '+SWRD3;
            D:=DCHARS[1];IF D IN ['A'..'Z'] THEN DCHARST:=SWRD2
               ELSE
                  BEGIN
                    SCHARS1:=' '+DCHARS;LEN:=LEN+LENGTH(SCHARS1);
                    GOTO L2;
                  END;
            CREATE_ALL_ARR;
            GOTO L3;
       END;
    SCHARS1:=' '+SWRD1;LEN:=LEN+LENGTH(SCHARS1);STORE_IN_LINE;
    DWRD := COPY(TWRD,LENGTH(SWRD1)+2,LENGTH(TWRD)-LENGTH(SWRD1)+1);
    D:=DWRD[1];IF D  IN ['A'..'Z'] THEN CREATE_ALL_ARR
         ELSE BEGIN SCHARS1:=' '+DWRD ;LEN:=LEN+LENGTH(SCHARS1);
         GOTO L2; END;
    GOTO N3;
END;


BEGIN
 LHD:=NIL;{WRITELN('MEM AVAIL =',MEMAVAIL);DELAY(2000);}
 FLAG:=FALSE;FLAG1:=TRUE;FLAG2:=TRUE;EDTR:=TRUE;PRMT:=FALSE;
 LEN:=0;KEYCNT:=0;
 T1:WINDOW(1,1,80,25);CLRSCR;
 GOTOXY(27,12);WRITELN('1: CREATE YOUR FIRST DICTIONARY');
 GOTOXY(27,13);WRITELN('2: UPDATE EXISTING DICTIONARY');
 GOTOXY(27,14);WRITELN('3: DISPLAY DICTIONARY');
```

```
GOTOXY(27,15);WRITELN('4: EDIT TEXT');
GOTOXY(27,16);WRITELN('5: PRINT FILE');
GOTOXY(27,17);WRITELN('6: EXIT' );
   GOTOXY(20,20);
   WRITE('PLEASE ENTER YOUR CHOICE (1,2,3,4,5,6) : ');READ(KBD,CI);
   WRITELN(CI);
   CASE CI OF
     '1':CREATE_FST_DIC;
     '2':UPDATE_DIC;
     '3':PRINT_ALL_DICS;
     '4':BEGIN
         L11:  EDITOR;
               IF QUIT_EDIT THEN BEGIN FLAG1:=TRUE;GOTO TT1;END;
               EDIT_TEXT;
               IF QUIT_EDIT THEN
                BEGIN
                  GOTOXY(1,23); WRITE('FILE SAVE (Y/N) ? ');
                  READ(KBD,C);
                  IF (C='Y') OR (C ='y') THEN SAVE_TEXT;
                  FLAG1:=TRUE;GOTO TT1;
                  END;
               GOTO L11;
         END;
      '5':PRINT_FILE;
      '6':EXIT;
    END;
TT1: GOTO T1;
END.
```