DESIGN & DEVELOPMENT OF A NEW AND EFFICIENT APPROACH FOR LINE TEXT EDITING IN TURBO C FOR WINDOWS*

K. J. Satao**

ABSTRACT

Text editors come in the forms viz. Line editors, Stream editors, Screen editors, Word processors, Structure editors, etc. There are many text editors provided with Windows viz. Notepad, WordPad, Microsoft Office Word, etc. The MS-DOS editors viz. EDLIN, EDIT also work in Windows. But none of the above editors is interactive. The process of entering the programs/text shall be greatly enhanced, for the newcomers, if the editor is an interactive one. This paper gives the design & development of a new and efficient approach for line text editing, the most basic and fundamental editing, in Turbo C for Windows OS. It is proposed to give a new, interactive, and more user friendly approach to line text editing for document / non document(program) files using singly linked lists. A menu driven program is designed and developed. The program displays all the options and once a user chooses a particular option, he is prompted further interactively. The program is fully tested on Windows XP, with total RAM = 256 MB. This research work has, in all, twenty three options and is academically(not available in any book) very useful for the Computer Science & Engineering and Information Technology disciplines.

Keywords : Authoring tools and methods; Computer-mediated communication; Human-computer interface; Interactive learning environments; Programming and programming languages.

** Professor & Head, Computer Science and Engineering, Rungta College of Engg. & Technology, Kohka Road, Kurud, Bhilai-490 024, Chhattisgarh, INDIA.

1. INTRODUCTION

The options in the research work are…"C" for clearing the screen, "A" for appending lines, "L" for listing from a line number to a line number, "E" for listing the entire file, "I" for inserting lines before a line number, "D" for deleting from a line number to a line number, "U" for updating a line, "W" for writing entire text in the file opened or currently created by the user, "F" for writing specific lines in a specific file, "O" for copying from a line number to a line number before a line number, "M" for moving from a line number to a line number before a line number, "S" for searching a string, "R" for replacing a string, "G" for listing page wise(desired number of lines at a time), "V" for converting upper case to lower case and vice-versa or to title/sentence case, from a line number to a line number, "T" for copying from another file before a line number, "P" for printing a file with or without heading, "Y" for displaying directory contents, "K" for deleting files, "J" for searching files in a directory for a string, "H" for displaying or modifying file attributes, "N" for opening a new file, and "Q" for quitting.

This research work has the following main features...

1. No user training is required. A user can use FDD / HDD / CD-ROM drive / Pen drive.
2. No referring of the manual for the options is required. The program gives starting and ending times.
3. It is very useful in the students' environment i.e. in the schools and colleges where a batch of students may come for the practical work and a single instructor/teacher is supposed to look after the batch.
4. This work is compatible with text processing utilities such as EDLIN, EDIT, Notepad, etc.
5. A file can be created in or saved in or loaded from any directory. Program displays file attributes.
6. Insertion/appending/updation is made very easy. The program takes care of RO, H, S, D, A, Vol Id files.
7. Each option gives a starting message so that the user becomes sure about what he shall be doing and an ending message so that he shall become sure about the completion of the desired task.
8. A user shall be able to update files even with the .bak extension. A backup file with an extension of .OLD is created before every overwriting of the file.

* Orally presented and demonstrated in the National Conference on "Advanced Computing Techniques", "BITCON – 2007", organized by Bhilai Institute of Technology, Durg, C.G., India, held during 16th & 17th March 2007.

2. IMPORTANT FUNCTIONS AND MESSAGES INCORPORATED

[Modified, Windows, C versions of Satao, 1992 and 1994]

2.1 Function usage()

```
'An  Interactive Line Text Editor for  Windows  using Turbo C  By Prof. K.J.Satao'
'Usage : WCEDITOR<Enter> from the prompt or'
'        Double Click on WCEDITOR Icon or'
'        Select WCEDITOR Icon and Hit Enter key'
'(Alt & Enter keys give full screen view, pressing again gives title bar)'
```

2.2 Function get_drive_name()

```
'Please enter Floppy/Hard Disk/CD-ROM/Pen drive name...A/B/C/D/E/F/G/H etc.'
'(Pl.Note : If the drive does not exist then the default drive is considered with'
' it's root/current directory) '
```

2.3 Function get_file_name_message()

```
'Please give file name...'
'You may enter just file name(8+3) for current directory'
'Or you may  enter \file name(8+3) for root directory file'
'Or give path & file name e.g. TC\BIN\WCEDITOR.C'
'78 characters maximum(case insensitive e.g. TeST.c = TEST.C) '
```

2.4 Function dis_mem_cap() { Display Memory Capacity }

```
'Main memory available = ', coreleft, ' Bytes'
```

2.5 Function get_file_name2()

```
get_file_name_message()
'You may hit just Enter key for *.* in current directory'
'You can use *, ? as wild characters(? for single character). '
```

2.6 Welcome Message

```
'File:FILE_NAME                              Date:DD/MM/YYYY Day:DAY Time:HH:MM:SS'
'-------------------------------------------------------------------------------'
'Welcome to An Interactive Line Text Editor by Prof. K. J. Satao,Bhilai(CG),India'
'-------------------------------------------------------------------------------'
'Please choose an option...C-Clear screen/A-Append/L-List specific lines/E-Entire'
'list/I-Insert/D-Delete/U-Update/W-Write in open file/F-Write in any file/O-Copy/'
'M-Move/G-Pagewise list/V-Convert case/T-Copy from other file/S-Search/R-Replace/'
'P-Print a file/Y-Directory/K-Delete files/H-Change attributes/J-Search in files/'
'N-Open new file/Q-Quit(Upper/Lower Case) '
```

2.7 Function number_check()

```
if (number = 0) printf 'Error in number, Value = 0, New value = 1'
if (number < 0) printf 'Error in number, Value is negative, New value = 1'
```

2.8 Function append_insert_update_message()

```
'Use F1 or --> to take a character from the  previous  line in the current line.'
'Use F2 or <-- or Del to delete a character from previous line for current line.'
'(F2 or <-- or Del does not  physically  delete a character from previous line).'
'Use F3 or End to take  all  the  remaining  characters from the  previous line.'
'Tab key inserts 5 blank characters. Backspace key  deletes  previous character.'
```

2.9 Insert_Append Message

```
'You can have up to 132 characters per line. End each line with Enter key.'
'New line begins automatically after 132 characters are entered in a line.'
'Use Ctrl d/D to end appending/inserting. You can have maximum 3617 lines.'
append_insert_update_message()
'Save(W/F) your work at regular intervals to avoid the accidental loss of data.'
```

2.10 Update Message

```
'You can have  maximum  132 characters in the line. End the line with Enter key.'
append_insert_update_message()
'Updation  automatically  stops after 132 characters are  entered  in the line.'
```

2.11 Function get_lines()

```
'From which line number ? '(Say number)
'To   which line number ? '(Say stop)
```

2.12 Delete Message

```
'Please note...this option will delete the desired lines.'
'The deleted lines cannot be recovered later on unless the lines are written in a'
'file before proceeding.'
'Do you surely want to delete line(s) ? (Y/y for yes) '
```

2.13 Convert Case Message

```
'Please enter...L/l for lower case or U/u for upper case or T/t for title case or'
'S/s for sentence case or Hit Enter key to abort '
```

2.14 Function search_options()

```
'Do you want to ignore case ? (Y/y for yes) '
'? can be used as a wild character e.g. s???o shall match with satao, spqro, etc.'
'Be careful in using it...Use it ? (Y/y for yes) '
'Which string ? (First 20 characters are considered...Hit just Enter to abort)'
```

2.15 Replace Message

```
search_options()
'By which string ? (First 20 characters are considered)'
'Replace by query ? (Y/y for yes) '
```

2.16 Function print_file()

```
'Printing a file on parallel printer i.e. LPT1/PRN'
get_file_name1()
'Please save your work before proceeding...Runtime error may occur...'
'Choose...1..To print or 2..To abort '(Say i)
'Choose...1..For 80 column printer or 2..For 132 column printer '(Say j)
'Do you want page heading ? (Y/y for yes) '(Say ch1)
```

2.17 Function new_file

```
'Please note...this option will delete all current lines.'
'The deleted lines cannot be recovered later on.'
'You should proceed only after writing the current lines in a file.'
'Do you surely want to proceed ? (Y/y for yes) '
```

3. PROPOSED FURTHER RESEARCH WORK

It is proposed to use doubly linked lists, circular linked lists and develop interactive line text editors and screen editors. It is proposed to remove the present limitations of the program viz. Windows uses up to 256 characters file name whereas the current program uses 8+3 format, Windows mostly uses USB printer whereas the program uses parallel printer.

4. ACKNOWLEDGEMENTS

The author is highly thankful to Hon'ble Shri Santosh Rungta, Respected Shri Sourabh Rungta, and Respected Shri Sonal Rungta of Rungta College of Engineering & Technology, Bhilai for providing the necessary facilities for the work.

5. REFERENCES

1. Satao K. J., 1992, An Interactive Line Text Editor for MS-DOS Operating System, Indian Computing Congress, 1992, Hyderabad, India, held during Dec. 17-19, 1992; published in the proceedings of the conference, ISBN : 0-07-462029-0, titled "Indian Computing Congress Series - Innovative Applications in Computing" edited by E. Balgurusamy & B. Sushila, published by Tata McG Hill, at P. P. 439-460.

2. Satao K. J., 1994, An Interactive Line Text Editor for Novell's Netware Operating System, National Conference - "Frontier'94" at Babasaheb Naik College of Engineering, Pusad(M.S.), India, held during Feb. 1-2, 1994; published in the proceedings of the conference at P. P. 131-145.

APPENDIX : THE SOURCE CODE OF THE PROGRAM

[Modified, Windows, C version of Satao, 1992 and 1994]

```
/* Design and Development of a New and Efficient  Approach(Interactive) for Line *
 * Text Editing using Linked Lists in C Language for Windows O.S.,WCEDITOR.C, by *
 * Prof. K.J.Satao, Rungta College of Engineering & Technology,Kohka Road,Kurud, *
 * Bhilai-490 024,C.G.,India. Compiler used is Turbo C++, Version 3.0. Tested on *
 * Microsoft  Windows  XP Professional with  Total  Physical  Memory  =  256 MB, *
 * Available Physical Memory = 35.93 MB. Build using the  Large Memory Model and *
 * run the executable program. For  Memory  Model...Choose  Alt O from the IDE : *
 * Options - Compiler - Code generation... - Model - Large. Final version.       */
#include "stdio.h"
#include "conio.h"
#include "stdlib.h"
#include "dos.h"
#include "io.h"
#include "errno.h"
#include "alloc.h"
#include "string.h"
#include "time.h"
#define TRUE 1
#define FALSE 0
char reserved_file_names[13][7] =
 { "AUX","CLOCK$","COM1","COM2","COM3","COM4",
   "CON","LPT1","LPT2","LPT3","LST","NUL","PRN" };
char day_name[7][4] =
 { "SUN","MON","TUE","WED","THU","FRI","SAT" };
char allowed_file_name_chars[83] =
 { '0','1','2','3','4','5','6','7','8','9',
   'A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S',
   'T','U','V','W','X','Y','Z','a','b','c','d','e','f','g','h','i','j','k','l',
   'm','n','o','p','q','r','s','t','u','v','w','x','y','z','*','\?','(',')','_',
   '^','$','~','!','#','%','&','-','{','}','@','\'','.','\\',' ' };
char word_separators[33] =
 { '`','~','!','@','#','$','%','^','&','*','(',')','-',
   '_','+','=','{','[','}',']','|','\\',':',';','\"',',','<','.','>','/','\?',' ' };
char sentence_separators[10] =
 { '!','@','(','{','[',':','<','.','\?' };
char dn1[_MAX_DRIVE], d1[_MAX_DIR], n1[_MAX_FNAME], e1[_MAX_EXT], dr_name;
typedef char text1[134];          //Maximum line length = 132 chars + LF + '\0'
typedef char string11[_MAX_PATH]; //Path 80 characters + '\0'
string11 file_name, old_file_name, f_name, asciiz, s1, o_asciiz;
typedef char string12[21];        //Search/replace 20 characters at a time + '\0'
struct linked_list_1              //First linked list for main text
{ int line_no;
  text1 line_text;
  struct linked_list_1 *next;
};
struct linked_list_2              //Second linked list for copy/move lines
{ int line_no1;
  text1 line_text1;
  struct linked_list_2 *next1;
};
text1 new_line, new_line1, temp, temp1, temp2, b; int linenum, pgno;
string12 temp12, temp14, b12, b13, rep_string; struct ftime ft; FILE *stream;
```

```
typedef struct linked_list_1 node; node *head, *link, *prev, *p;
typedef struct linked_list_2 node1; node1 *head1, *link1, *prev1, *p1;
struct ffblk ffblk; struct dosdate_t d; struct dostime_t t; char ch2[3];
int number, start, stop, highest_no, highest_no1, line_occ, fval, sno, line_rep;
char ignore_case, query, wild, option, choice, ch, ch1, old_ch, over_flow, a[6];
long int tot_ch, tot_occ, rep_done, tot_ch_cp; FILE *f, *fp, *from_file, *to_file;
int done, saved, m, n, i, j, k, i1, j1, k1, nu, nu1, occ, l, l1, sen_sep;
int valid_file_name, dr_no, available_memory; char buf[82]; char stdin_str[256];
int hours1,minutes1,seconds1,hours2,minutes2,seconds2,hours11,minutes11,seconds11;
int page_heading_chars, page_heading_lines, line_occ1, t_ch;
void main() //Starting of main function
{ date_time(); hours1 = t.hour; minutes1 = t.minute; seconds1 = t.second;
  textbackground(RED); setcbrk(FALSE); new_file();
  do
  { l1: date_time(); textcolor(YELLOW); cprintf("\r\nFile:");
    textcolor(CYAN); cprintf("%s ", file_name); textcolor(YELLOW); cprintf("Date:");
    textcolor(CYAN); cprintf("%02d/%02d/%04d ", d.day, d.month, d.year);
    textcolor(YELLOW); cprintf("Day:"); textcolor(CYAN);
    cprintf("%s ", day_name[d.dayofweek]); textcolor(YELLOW); cprintf("Time:");
    textcolor(CYAN); cprintf("%02d:%02d:%02d\r\n", t.hour, t.minute, t.second);
    textcolor(YELLOW);
    cprintf("------------------------------------------------------------------"
            "----------------"); textcolor(CYAN);
    cprintf("Welcome to An Interactive Line Text Editor by Prof. K. J. Satao"
            ",Bhilai(CG),India"); textcolor(YELLOW);
    cprintf("------------------------------------------------------------------"
            "----------------");
    cprintf("Please choose an option...");
    textcolor(CYAN);cprintf("C");textcolor(YELLOW);cprintf("-Clear screen/");
    textcolor(CYAN);cprintf("A");textcolor(YELLOW);cprintf("-Append/");
    textcolor(CYAN);cprintf("L");textcolor(YELLOW);cprintf("-List specific lines/");
    textcolor(CYAN);cprintf("E");textcolor(YELLOW);cprintf("-Entirelist/");
    textcolor(CYAN);cprintf("I");textcolor(YELLOW);cprintf("-Insert/");
    textcolor(CYAN);cprintf("D");textcolor(YELLOW);cprintf("-Delete/");
    textcolor(CYAN);cprintf("U");textcolor(YELLOW);cprintf("-Update/");
    textcolor(CYAN);cprintf("W");textcolor(YELLOW);cprintf("-Write in open file/");
    textcolor(CYAN);cprintf("F");textcolor(YELLOW);cprintf("-Write in any file/");
    textcolor(CYAN);cprintf("O");textcolor(YELLOW);cprintf("-Copy/");
    textcolor(CYAN);cprintf("M");textcolor(YELLOW);cprintf("-Move/");
    textcolor(CYAN);cprintf("G");textcolor(YELLOW);cprintf("-Pagewise list/");
    textcolor(CYAN);cprintf("V");textcolor(YELLOW);cprintf("-Convert case/");
    textcolor(CYAN);cprintf("T");textcolor(YELLOW);cprintf("-Copy from other file/");
    textcolor(CYAN);cprintf("S");textcolor(YELLOW);cprintf("-Search/");
    textcolor(CYAN);cprintf("R");textcolor(YELLOW);cprintf("-Replace/");
    textcolor(CYAN);cprintf("P");textcolor(YELLOW);cprintf("-Print a file/");
    textcolor(CYAN);cprintf("Y");textcolor(YELLOW);cprintf("-Directory/");
    textcolor(CYAN);cprintf("K");textcolor(YELLOW);cprintf("-Delete files/");
    textcolor(CYAN);cprintf("H");textcolor(YELLOW);cprintf("-Change attributes/");
    textcolor(CYAN);cprintf("J");textcolor(YELLOW);cprintf("-Search in files/");
    textcolor(CYAN);cprintf("N");textcolor(YELLOW);cprintf("-Open new file/");
    textcolor(CYAN+BLINK); cprintf("Q"); textcolor(YELLOW);
    cprintf("-Quit(Upper/Lower Case) ");
    sound1(); flushall(); option = toupper(getche()); cprintf("\r\n\r\n");
    if ((option <= '?') || (option >= 'Z') || (option == '@') ||
        (option == 'B') || (option == 'X') || (option == '\0'))
    { textcolor(CYAN); cprintf("\r\nRetry...\r\n\r\n"); textcolor(YELLOW); goto l1;}
```

```
    switch(option)
    { case 'C':clear_screen();
        break;
      case 'E':cprintf("List entire file...E/e\r\nHit any key to continue ");
        sound1(); getch(); cprintf("\r\n\r\n");
        number = 1; stop = highest_no - 1;
        list_lines(number, stop);
        cprintf("\r\nListing over...Total characters including CR,LF = %ld"
                "\r\nHit any key to continue ", tot_ch + stop - number + 1);
        sound1(); getch(); cprintf("\r\n");
        break;
      case 'I':cprintf("Insert lines...I/i\r\n\r\nBefore which line number ? ");
        flushall(); sound1(); gets(a); number = atoi(a);
        number_check(number);
        if (number < 1) number = 1;
        if (number > highest_no) //Insert required blank lines
        { cprintf("\r\n");
          insert_required_blank_lines(number);
          if (!(available_memory)) goto l5;
        }
        cprintf("\r\n"); insert_append();
l5:     cprintf("\r\n\r\nTotal lines = %d, Insertion over..."
                "Hit any key to continue ", highest_no - 1);
        sound1(); getch(); cprintf("\r\n");
        dis_mem_cap(); saved = FALSE;
        break;
      case 'A':cprintf("Append lines...A/a\r\n\r\nStarting appending...\r\n\r\n");
        number = highest_no; insert_append();
        cprintf("\r\n\r\nAppending over...Hit any key to continue ");
        sound1(); getch(); cprintf("\r\n");
        dis_mem_cap(); saved = FALSE;
        break;
      case 'D':cprintf("Delete lines...D/d\r\n");
        if (highest_no > 1)
        { cprintf("\r\nPlease note...this option will delete the desired lines."
                  "\r\nThe deleted lines cannot be recovered later on unless the "
                  "lines are written in afile before proceeding."
                  "\r\n\r\nDo you surely want to delete line(s) ? (Y/y for yes) ");
          sound1(); ch = toupper(getche()); cprintf("\r\n");
          if (ch == 'Y')
          { get_lines();
            for (i = number; i <= stop; i++) delete_line(i);
            highest_no -= (stop - number + 1);
            prev = head; link = prev -> next; nu1 = stop - number + 1;
            while (link != NULL)
            { nu = link -> line_no; if (nu > stop) nu -= nu1;
              link -> line_no = nu; link = link -> next;
            }
            cprintf("\r\nTotal lines remaining = %d, Deletion over..."
                    "Hit any key to continue ", highest_no - 1);
            sound1(); getch(); cprintf("\r\n");
            dis_mem_cap(); saved = FALSE;
          } else use_w_or_f();
        } else zero_lines();
        break;
```

```
      case 'L':cprintf("List from a line number to a line number...L/l\r\n");
        if (highest_no > 1)
        { get_lines();
          cprintf("\r\n"); list_lines(number, stop);
          cprintf("\r\nListing over...Total characters including CR,LF = %ld"
                  "\r\nHit any key to continue ", tot_ch + stop - number + 1);
          sound1(); getch(); cprintf("\r\n");
        } else zero_lines();
        break;
      case 'O':cprintf("Copy & Paste lines...O/o\r\n");
        if (highest_no > 1)
        { get_lines();
          cprintf("\r\nPaste before which line number ? ");
          flushall(); sound1(); gets(a); sno = atoi(a); number_check(sno);
          if (sno < 1) sno = 1;
          if (sno > highest_no) //Insert required blank lines
          { insert_required_blank_lines(sno); if (!(available_memory)) goto l10; }
          copy_lines(number, stop); l1 = sno; link1 = head1 -> next1;
          while (link1 != NULL)
          { //Insert lines in main text from second linked list
            strncpy(temp, link1 -> line_text1, 134); strncpy(new_line1, temp, 134);
            insert_line(l1, new_line1); if (!(available_memory)) goto l10;
            highest_no++; l1++; link1 = link1 -> next1;
          }
l10:      cprintf("\r\nTotal lines = %d, Copying & Pasting over..."
                  "Hit any key to continue ", highest_no - 1);
          sound1(); getch(); cprintf("\r\n");
          dis_mem_cap(); head1 = (node1 *)malloc(sizeof(node1));
          head1 -> next1 = NULL; free(head1); /*Free second linked list*/
          saved = FALSE;
        } else zero_lines();
        break;
      case 'M':cprintf("Move lines(Cut & Paste)...M/m\r\n");
        if (highest_no > 1)
        { get_lines();
          cprintf("\r\nPaste before which line number ? ");
          flushall(); sound1(); gets(a); sno = atoi(a); number_check(sno);
          if (sno < 1) sno = 1;
          if (sno > highest_no) //Insert required blank lines
          { insert_required_blank_lines(sno); if (!(available_memory)) goto l15; }
          if ((sno >= number) && (sno <= stop)) continue;
          copy_lines(number, stop); link1 = head1 -> next1;
          for (l1 = sno; l1 <= (sno + stop - number); l1++)
          { //Insert lines in main text from second linked list
            strncpy(temp, link1 -> line_text1, 134); strncpy(new_line1, temp, 134);
            insert_line(l1, new_line1); if (!(available_memory)) goto l15;
            highest_no++; link1 = link1 -> next1;
          }
          cprintf("\r\n");
          if (sno < number)
          { start = number + (stop - number + 1);
            stop = stop + (stop - number + 1);
          }
          else start = number;
          for (i = start; i <= stop; i++) delete_line(i); //Delete copied lines
          highest_no -= (stop - start + 1); prev = head;
```

```
          link = prev -> next; nu1 = stop - start + 1;
          while (link != NULL)
          { nu = link -> line_no; if (nu > stop) nu -= nu1;
            link -> line_no = nu; link = link -> next;
          }
l15:      cprintf("Cut & Paste over...Hit any key to continue ");
          sound1(); getch(); head1 = (node1 *)malloc(sizeof(node1));
          head1 -> next1 = NULL; cprintf("\r\n");
          free(head1); /* Free second linked list */ saved = FALSE;
        } else zero_lines();
        break;
      case 'U':cprintf("Update a line...U/u\r\n\r\nWhich line number ? ");
        flushall(); sound1(); gets(a); number = atoi(a); number_check(number);
        cprintf("\r\n"); if (number < 1) number = 1;
        if (!(number > highest_no - 1))
        { textcolor(CYAN);
          cprintf("You can have up to 132 characters in the line. "
                  "End the line with Enter key.\r\n");
          append_insert_update_message();
          cprintf("Updation automatically stops after 132 characters "
                  "are entered in the line.\r\n\r\n");
          textcolor(YELLOW); list_lines(number, number); flushall();
          cprintf("%4d:",number);l = -1; for (i = 0; i <= 133; i++) temp1[i] = '\0';
          insert_append1(); delete_line(number); insert_line(number, new_line);
          if (!(available_memory)) goto l20; sound1();
          cprintf("\r\n\r\nUpdation over...Hit any key to continue "); getch();
        } else
        { textcolor(CYAN);
          cprintf("\r\nInvalid number...Maximum line number existing is...%d\r\n"
                  "\r\nUpdation aborted...Hit any key to continue ", highest_no-1);
          textcolor(YELLOW); sound1(); getch();
        } cprintf("\r\n"); dis_mem_cap(); saved = FALSE;
l20:    break;
      case 'P':cprintf("Printing a file on the printer...P/p\r\n\r\n");
        print_file();
        break;
      case 'G':cprintf("Pagewise list...G/g\r\n\r\n");
        if (highest_no > 1)
        { cprintf("Howmany lines per page ? (Maximum 22) ");
          flushall(); sound1(); gets(a); number = atoi(a); number_check(number);
          if (number > 22) number = 22; if (number < 1) number = 1;
          i = 1; stop = highest_no - 1;
          if (number == 1)
          { cprintf("\r\nHit any key to continue "); sound1(); getch(); }
          do
          { fval = i + number - 1; clear_screen();
            if (fval >= stop) fval = stop; list_lines(i, fval); i = ++fval;
            flushall(); cprintf("\r\n"); textcolor(CYAN);
            cprintf("Enter character S/s to stop,any other character to continue ");
            textcolor(YELLOW); sound1(); ch = toupper(getche());
          } while ((fval <= stop) && (ch != 'S'));
          cprintf("\r\n\r\nDisplay over...Hit any key to continue ");
          sound1(); getch(); cprintf("\r\n");
        } else zero_lines();
        break;
```

```
case 'F':cprintf("Write specific lines to a specific file...F/f\r\n");
  if (highest_no > 1)
  { get_lines(); get_check_file_name1();
    write_lines_in_file(); dis_file_names();
  } else zero_lines();
  break;
case 'S':cprintf("Search a string...S/s\r\n");
  if (highest_no > 1)
  { cprintf("\r\n"); search_options();
    if (!(l == 0))
    { number = 1; stop = highest_no - 1; search_string(number, stop, b12);
    } else
    { textcolor(CYAN); cprintf("String length = 0...So no searching");
      textcolor(YELLOW); cprintf("\r\nHit Enter key to continue ");
      sound1(); hit_enter();
    }
  } else zero_lines();
  break;
case 'R':cprintf("Replace a string...R/r\r\n");
  if (highest_no > 1)
  { cprintf("\r\n"); search_options();
    if (!(l == 0))
    { for (i = 0; i <= 20; i++) temp12[i] = '\0';
      cprintf("By which string ? (First 20 characters are considered)\r\n");
      sound1(); fgets(stdin_str, sizeof(stdin_str), stdin);
      if (strlen(stdin_str) > 21)
      { textcolor(CYAN);
        cprintf("Number of characters > 20, "
                "First 20 characters are considered\r\n");
        textcolor(YELLOW);
      }
      strncpy(rep_string, stdin_str, sizeof(rep_string));
      rep_string[strlen(rep_string) - 1] = '\0'; m = strlen(rep_string);
      rep_string[m] = '\0'; m = strlen(rep_string);
      for (i = 0; i <= m - 1; i++) temp12[i] = rep_string[i];
      strncpy(b13, temp12, 21); sound1();
      cprintf("\r\nReplace by query ? (Y/y for yes) "); hit_enter();
      query = toupper(ch2[0]); number = 1; stop = highest_no - 1;
      replace_string(number, stop, b12, b13);
    } else
    { textcolor(CYAN); cprintf("String length = 0...So no replacement");
      textcolor(YELLOW); cprintf("\r\nHit Enter key to continue ");
      sound1(); hit_enter();
    } saved = FALSE;
  } else zero_lines();
  break;
case 'T':cprintf("Copy & Paste lines from another file...T/t\r\n");
  strncpy(old_file_name, file_name, 80);
  get_check_file_name1(); f = fopen(file_name, "r"); fclose(f);
  if (f == NULL)
  { textcolor(CYAN);
    cprintf("\r\nFile does not exist/Wrong path...So no copying"
            "\r\nHit any key to continue ");
    sound1(); getch(); textcolor(YELLOW);
    cprintf("\r\n"); strncpy(file_name, old_file_name, 80); continue;
  }
```

```
        cprintf("\r\nPaste before which line number ? ");
        flushall(); sound1(); gets(a); number = atoi(a);
        number_check(number); if (number < 1) number = 1;
        if (number > highest_no) //Insert required blank lines
        { insert_required_blank_lines(number); if (!(available_memory)) goto l25;
        }
        read_lines_from_file(number); textcolor(CYAN);
        if (!(available_memory)) cprintf("\r\nFile truncated...\r\n\r\n");
        textcolor(YELLOW);
l25:    cprintf("Total lines = %d, Copying & Pasting from another file over...\r\n"
                "Hit any key to continue ", highest_no - 1);
        sound1(); getch(); cprintf("\r\n"); dis_mem_cap(); saved = FALSE;
        strncpy(file_name, old_file_name, 80);
        break;
      case 'W':cprintf("Write entire file...W/w\r\n");
        if (highest_no > 1)
        { cprintf("Writing in open file...%s\r\n", file_name);
          stop = highest_no-1; number = 1; write_lines_in_file();
          dis_file_names(); saved = TRUE;
        } else zero_lines();
        break;
      case 'V':cprintf("Case convert...N/n\r\n");
        if (highest_no > 1)
        { get_lines();
          textcolor(CYAN);
          cprintf("\r\nPlease enter...L/l for lower case or U/u for upper case or "
                  "T/t for title case orS/s for sentence case or any other "
                  "character to abort ");
          flushall(); sound1(); choice = toupper(getche()); textcolor(YELLOW);
          if ((choice != 'L') && (choice != 'U') &&
              (choice != 'T') && (choice != 'S'))
          { cprintf("\r\nChoice is not L/l or U/u or T/t or S/s..."
                    "\r\nHence...Case is not changed...Hit any key to continue ");
            sound1(); getch(); cprintf("\r\n"); continue;
          }
          cprintf("\r\n");
          if (choice == 'L') lower_case(number, stop);
          else if (choice  == 'U') upper_case(number, stop);
          else if ((choice == 'T') || (choice == 'S'))
          { lower_case(number, stop); title_sentence_case(number, stop); }
          saved = FALSE; cprintf("\r\nCase is converted...Hit any key to check ");
          sound1(); getch(); cprintf("\r\n\r\n"); list_lines(number, stop);
        } else zero_lines();
        break;
      case 'Y':cprintf("Directory...Y/y\r\n");
        strncpy(old_file_name,file_name,80);
        display_directory(); strncpy(file_name, old_file_name, 80);
        break;
      case 'J':cprintf("Search file(s) in a directory for a string ...J/j");
        cprintf("\r\n"); strncpy(old_file_name, file_name, 80);
        search_files(); strncpy(file_name, old_file_name, 80);
        break;
      case 'H':cprintf("Change file attribute(s)...H/h\r\n");
        set_file_attributes(); dis_open_file();
        break;
```

```
      case 'K':cprintf("Delete one or more archive files...K/k\r\n");
        strncpy(old_file_name, file_name, 80); delete_files();
        cprintf("\r\nHit any key to check ");
        sound1(); getch(); cprintf("\r\n");
        display_directory(); strncpy(file_name, old_file_name, 80); dis_open_file();
        break;
      case 'N':cprintf("Opening a new file...N/n\r\n\r\n"
                       "Please note...this option will delete all current lines."
                       "\r\nThe deleted lines cannot be recovered later on."
                       "\r\nYou should proceed only after writing"
                       " the current lines in a file.\r\n"
                       "\r\nDo you surely want to proceed ? (Y/y for yes) ");
        sound1(); ch = toupper(getche());
        if (ch == 'Y') new_file(); else { cprintf("\r\n"); use_w_or_f(); }
        break;
      case 'Q':cprintf("Quit...Q/q\r\n\r\n");
        if (saved)
        { head = (node *)malloc(sizeof(node));
          head -> next = NULL; free(head); /* Free first linked list */ goto l2;
        }
        else
        { textcolor(CYAN);
          cprintf("Text is possibly changed but not saved\r\n\r\n"
                  "Do you want to save it(write in a file) ? (Y/y for yes) ");
          flushall(); sound1(); ch = toupper(getche()); cprintf("\r\n\r\n");
          textcolor(YELLOW);
          if (ch == 'Y')
          { use_w_or_f(); option = 'C'; continue;
          } else
          { head = (node *)malloc(sizeof(node));
            head -> next = NULL; free(head); /* Free first linked list */ goto l2;
          }
        }
       } //End of switch
     } while (option != 'Q');
l2:  textcolor(CYAN); date_time();
     hours2 = t.hour; minutes2 = t.minute; seconds2 = t.second;
     cprintf("Starting time(HH:MM:SS) : %2d:%2d:%2d\r\n"
             "Ending   time(HH:MM:SS) : %2d:%2d:%2d\r\n",
             hours1, minutes1, seconds1, hours2, minutes2, seconds2);
     if (seconds2 < seconds1)
     { seconds11 = seconds2 + 60 - seconds1; minutes1++; }
     else seconds11 = seconds2 - seconds1;
     if (minutes2 < minutes1)
     { minutes11 = minutes2 + 60 - minutes1; hours1++; }
     else minutes11 = minutes2 - minutes1;
     hours11 = hours2 - hours1; if (hours11 < 0) hours11 += 24;
     cprintf("\r\nElapsed  time(HH:MM:SS) : %2d:%2d:%2d\r\n"
             "\r\nSee you again...Bye..."
             "Hit any key ",
             hours11, minutes11, seconds11);
     sound1(); getch(); cprintf("\r\n");
     textcolor(WHITE); textbackground(BLACK);
} //End of main function
```

```
date_time()
{ _dos_getdate(&d); _dos_gettime(&t);
  return(0);
} //Date Time
sound1()
{ sound(1000 + random(2000)); delay(400); nosound();
  return(0);
} //Sound1
hit_enter()
{ fgets(stdin_str, sizeof(stdin_str), stdin);
  strncpy(ch2, stdin_str, sizeof(ch2)); ch2[strlen(ch2) - 1] = '\0';
  return(0);
} //Hit Enter
split_file_name()
{ for (i = 0; i <= strlen(file_name) - 1; i++)
  { ch = file_name[i];
    if ((ch >= 97) && (ch <= 122)) file_name[i] -= 32;
  } //Small letters are converted into capital letters
  fnsplit(file_name, dn1, d1, n1, e1);
  return(0);
} //Split file name : dn1=dr_name(2), d1=dir(67), n1=pri_name(8), e1=ext(4)(.+3)
test_file_name()
{ valid_file_name = TRUE; textcolor(CYAN);
  if (n1[0] == '\0')
  { cprintf("\r\nInvalid file name...Primary file name cannot be null\r\n");
    valid_file_name = FALSE; textcolor(YELLOW);
    return(0);
  }
  for (i = 0; i <= 12; i++)
  //Checking whether the primary file name is a reserved word or not
  if ((strcmp(n1, reserved_file_names[i]) == 0) && (e1[0] == '\0'))
  { cprintf("\r\nInvalid file name...It cannot be a reserved word\r\n");
    valid_file_name = FALSE; textcolor(YELLOW);
    return(0);
  }
  for (i = 0; i <= strlen(n1) - 1; i++)
  { valid_file_name = FALSE;  //Checking whether the primary file name contains
                              //the permitted characters or not
    for (j = 0; j <= 82; j++) //Checking if n1[i] is in allowed_file_name_chars
    if (n1[i] == allowed_file_name_chars[j]) { valid_file_name = TRUE; break; }
    if (valid_file_name == FALSE)
    { cprintf("\r\nInvalid file name...Invalid character... %c"
              " ...in primary file name\r\n", n1[i]);
      textcolor(YELLOW); return(0);
    }
  }
  if (e1[0] == '\0')
  { textcolor(YELLOW); return(0); }
  for (i = 1; i <= strlen(e1) - 1; i++)
  { valid_file_name = FALSE;
    //Checking whether the secondary file name contains the permitted
    //characters or not. Dot(.) is a part of extension, hence start with 1.
    for (j = 0; j <= 82; j++) //Checking if e1[i] in allowed_file_name_chars
    if (e1[i] == allowed_file_name_chars[j])
    { valid_file_name = TRUE; break; }
```

```
    if (valid_file_name == FALSE)
    { cprintf("\r\nInvalid file name...Invalid character... %c"
              " ...in secondary file name\r\n", e1[i]);
      textcolor(YELLOW); return(0);
    }
  } textcolor(YELLOW);
  return(0);
} //Test file name
display_file_attributes()
{ textcolor(CYAN);
  if ((ffblk.ff_attrib & FA_ARCH)   != 0) cprintf("<Archive>");
  if ((ffblk.ff_attrib & FA_RDONLY) != 0) cprintf("<Read Only>");
  if ((ffblk.ff_attrib & FA_HIDDEN) != 0) cprintf("<Hidden>");
  if ((ffblk.ff_attrib & FA_SYSTEM) != 0) cprintf("<System>");
  if ((ffblk.ff_attrib & FA_DIREC)  != 0) cprintf("<Directory>");
  if ((ffblk.ff_attrib & FA_LABEL)  != 0) cprintf("<Volume Id>");
  cprintf("\r\n"); textcolor(YELLOW);
  return(0);
} //Display file attributes
dis_open_file()
{ strncpy(f_name, file_name, 80); cprintf("\r\n");
  done = findfirst(file_name, &ffblk, -1);
  textcolor(CYAN); cprintf("Open file name = %s\r\n", file_name);
  textcolor(YELLOW);
  if (!(done))
  { cprintf("\r\nFile size = %ld Bytes, Attributes...", ffblk.ff_fsize);
    display_file_attributes();
  }
  return(0);
} //Display open file
dis_file_names()
{ fnsplit(file_name, dn1, d1, n1, e1); strcpy(asciiz, dn1);
  strcat(asciiz, d1); strcat(asciiz, n1); strcat(asciiz, "*.*");
  display_directory(); dis_open_file();
  return(0);
} //Display file names
dis_mem_cap()
{ textcolor(CYAN);
  cprintf("\r\nMain memory available = %lu Bytes\r\n", (unsigned long)coreleft());
  textcolor(YELLOW);
  return(0);
} //Display memory capacity
check_error()
{ i = _doserrno; j = 0; textcolor(CYAN); cprintf("\r\n");
  if (i == 3) cprintf("Error...Path not found\r\n\r\n");
  else if (i ==  6) cprintf("Error...Invalid handle\r\n\r\n");
  else if (i ==  8) cprintf("Error...Not enough memory\r\n\r\n");
  else if (i == 10) cprintf("Error...Invalid environment\r\n\r\n");
  else if (i == 11) cprintf("Error...Invalid format\r\n\r\n");
  textcolor(YELLOW);
  if ((i == 3) || (i == 6) || (i == 8) || (i == 10) || (i == 11))
  { cprintf("Retry...Hit any key to continue ");
    sound1(); getch(); cprintf("\r\n\r\n"); j = 1;
  }
  return(0);
} //Check error
```

```
get_path()
{ m = strlen(s1) - 1;
  strcpy(o_asciiz, asciiz);
  if ((strlen(asciiz) == 0) && (s1[m] != '\\'))
  { strcpy(asciiz, s1); strcat(asciiz, "\\*.*"); get_f_name(); return(0); }
  if ((strlen(asciiz) == 0) && (s1[m] == '\\'))
  { strcpy(asciiz, s1); strcat(asciiz, "*.*"); get_f_name(); return(0); }
  if ((d1[0] == '\0') && (s1[m] != '\\'))
  { strcpy(asciiz, s1); strcat(asciiz, "\\");
    strncat(asciiz, o_asciiz, strlen(o_asciiz)); get_f_name(); return(0);
  }
  if ((d1[0] == '\0') && (s1[m] == '\\'))
  { strcpy(asciiz, s1); strncat(asciiz, o_asciiz, strlen(o_asciiz));
    get_f_name(); return(0);
  }
  if ((d1[0] == '\\') && (s1[m] != '\\'))
  { for (i = 0; i <= 79; i++) asciiz[i] = '\0';
    asciiz[0] = dr_name; strcat(asciiz, ":");
    strncat(asciiz, o_asciiz, strlen(o_asciiz)); get_f_name(); return(0);
  }
  if ((d1[0] == '\\') && (s1[m] == '\\'))
  { strcpy(asciiz, s1); strcat(asciiz, n1); strcat(asciiz, e1);
    get_f_name(); return(0);
  }
  if (asciiz[strlen(asciiz)-1] == '\\')
  { strcat(asciiz, "*.*"); get_f_name(); return(0); }
  for (i = 0; i <= 79; i++) asciiz[i] = '\0';
  asciiz[0] = dr_name; strcat(asciiz, ":\\");
  strncat(asciiz, o_asciiz, strlen(o_asciiz)); get_f_name();
  return(0);
} //Get path
get_f_name()
{ strcpy(file_name, asciiz); split_file_name();
  test_file_name(); strcpy(asciiz, file_name);
  return(0);
} //Get file name
get_drive_name()
{ do
  { cprintf("\r\nPlease enter Floppy/Hard Disk/CD-ROM/Pen drive name...A/B/C/D"
            "/E/F/G/H etc.\r\n(Pl.Note : If the drive does not exist then the "
            "default drive is considered with it's root/current directory) ");
    sound1(); dr_name = toupper(getche());
  } while ((dr_name < 65) || (dr_name > 90));
  dr_no = dr_name - 64;
  return(0);
} //Get drive name
get_file_name_message()
{ cprintf("\r\nPlease give file name..."
          "\r\nYou may enter just file name(8+3) for current directory"
          "\r\nOr you may enter  \\file name(8+3) for root directory file"
          "\r\nOr give path & file name e.g. tc\\bin\\wceditor.c"
          "\r\n78 characters maximum(Case insensitive e.g. TeST.c = TEST.C)\r\n");
  return(0);
} //Get file name message
```

```
get_file_name1()
{ get_file_name_message(); sound1(); gets(asciiz);
  //File name cannot have \ | * ? < > +
  if ((strlen(asciiz) == 0))
  { textcolor(CYAN);
    cprintf("\r\nInvalid file name...File name cannot be null\r\n");
    valid_file_name = FALSE; textcolor(YELLOW);
    return(0);
  }
  for (i = 0; i <= strlen(asciiz) - 1; i++)
  if ((asciiz[i] == '*') || (asciiz[i] == '?'))
  { textcolor(CYAN);
    cprintf("\r\nInvalid file name...File name cannot have * or ?\r\n");
    valid_file_name = FALSE; textcolor(YELLOW);
    return(0);
  }
  fnsplit(asciiz, dn1, d1, n1, e1);
  _getdcwd(dr_no, buf, sizeof(buf));
  strcpy(s1, buf); get_path();
  return(0);
} //Get file name1
get_check_file_name1()
{l5:get_drive_name(); cprintf("\r\n");
  do get_file_name1(); while(!(valid_file_name));
  findfirst(file_name, &ffblk, -1); check_error();
  if (j == 1) goto l5;
  return(0);
} //Get check file name1
get_file_name2()
{ get_file_name_message();
  cprintf("\r\nYou may hit just Enter key for *.* in current directory\r\n"
          "You can use *, ? as wild characters(? for single character) " );
  //File name cannot have \ | < > +
  sound1(); flushall(); gets(asciiz);
  fnsplit(asciiz, dn1, d1, n1, e1);
  _getdcwd(dr_no, buf, sizeof(buf)); strcpy(s1, buf); get_path();
  return(0);
} //Get file name2
get_check_file_name2()
{l5:get_drive_name(); cprintf("\r\n");
  do get_file_name2(); while(!(valid_file_name));
  findfirst(file_name, &ffblk, -1); check_error();
  if (j == 1) goto l5;
  return(0);
} //Get check file name2
number_check(int number1)
{ textcolor(CYAN);
  if (number1 == 0)
  cprintf("Error in number, Value = 0, New value = 1\r\n");
  if (number1 < 0)
  cprintf("Error in number, Value is negative, New value = 1\r\n");
  textcolor(YELLOW);
  return(0);
} //Number check
```

```
append_insert_update_message()
{ cprintf("Use F1 or --> to take a character from the  previous  line "
          "in the current line.\r\n"
          "Use F2 or <-- or Del to delete a character from previous line "
          "for current line.\r\n"
          "(F2 or <-- or Del does not  physically  delete a character "
          "from previous line).\r\n"
          "Use F3 or End to take  all  the  remaining  characters "
          "from the previous line.\r\n"
          "Tab key inserts 5 blank characters. "
          "Backspace key  deletes previous character.\r\n");
  return(0);
} //Append insert update message
directory_heading()
{ if (option == 'Y') clear_screen();
  else cprintf("\r\n");
  cprintf("Directory : %s\r\n"
          "-------------------------------------------------------------"
          "--------------------"
          "S.N. File Name         Size  DD/MM/YYYY HH:MM:SS Attributes...\r\n"
          "-------------------------------------------------------------"
          "--------------------", asciiz);
  return(0);
} //Directory heading
display_directory()
{ int count;
  if (option == 'Y') get_check_file_name2();
  count = 0; done = findfirst(asciiz, &ffblk, -1);
  directory_heading();
  //All file types...RO, H, S, D, A, and Vol Id are listed
  while (!done)
  { textcolor(CYAN); count++;
    cprintf("%-5d%  12s %10ld ", count, ffblk.ff_name, ffblk.ff_fsize);
    strcpy(f_name, dn1); strcat(f_name, d1); strcat(f_name, ffblk.ff_name);
    stream = fopen(f_name, "rt"); getftime(fileno(stream), &ft);
    cprintf("%02u/%02u/%04u %02u:%02u:%02u ",
            ft.ft_day, ft.ft_month, ft.ft_year+1980, ft.ft_hour, ft.ft_min,
            ft.ft_tsec*2);
    display_file_attributes(); fclose(stream);
    i = count / 20;
    if (count == i * 20)
    { cprintf("-------------------------------------------------------------"
              "------Hit any key...");
      sound1(); getch(); textcolor(YELLOW);
      directory_heading();
    }
    done = findnext(&ffblk);
  }
  cprintf("-------------------------------------------------------------"
          "------Hit any key...");
  sound1(); getch(); textcolor(YELLOW);
  cprintf("\r\n\r\nTotal number of files = %d\r\n", count);
  return(0);
} //Display directory
```

```
set_file_attributes()
{ int attrib;
  strncpy(old_file_name, file_name, 80);
  get_check_file_name1();
  attrib = get_file_attrib(file_name); textcolor(CYAN);
  if (attrib == -1)
  switch(errno)
  { case ENOENT : cprintf("\r\nUnable to find...%s...Job aborted\r\n", file_name);
                  goto l7;
    case EACCES : cprintf("\r\nPermission denied...Job aborted\r\n");
                  goto l7;
    default     : cprintf("\r\nError number : %d...Job aborted\r\n", errno);
                  goto l7;
  } else
  { cprintf("\r\nFile attributes of...%s...are\r\n", file_name);
    if (attrib & FA_RDONLY) cprintf("<Read Only>");
    if (attrib & FA_HIDDEN) cprintf("<Hidden>");
    if (attrib & FA_SYSTEM) cprintf("<System>");
    if (attrib & FA_LABEL)  cprintf("<Volume Id>");
    if (attrib & FA_DIREC)  cprintf("<Directory>");
    if (attrib & FA_ARCH)   cprintf("<Archive>");
    cprintf("\r\n");
  } set_attributes();
l7:textcolor(YELLOW); cprintf("\r\nHit any key to recheck ");
  sound1(); getch(); cprintf("\r\n"); display_directory();
  strncpy(file_name, old_file_name, 80);
  return(0);
} //Set file attributes
get_file_attrib(char *file_name)  //Returns the attributes of a file
{ return(_chmod(file_name, 0)); } //Get file attributes
set_attributes()
{ unsigned attrib;
  textcolor(YELLOW);
  if (_dos_getfileattr(file_name, &attrib) != 0)
  { textcolor(CYAN); cprintf("\r\n\r\nUnable to obtain file attributes");
    textcolor(YELLOW); cprintf("\r\nHit any key to continue ");
    sound1(); getch(); cprintf("\r\n");
    return(0);
  }
  cprintf("\r\nDo you want to change the attributes ? (Y/y for yes) ");
  sound1();
  if (toupper(getche()) == 'Y')
  { textcolor(CYAN);
    if ((attrib & _A_VOLID) || (attrib & _A_SUBDIR))
    { textcolor(CYAN); cprintf("\r\n\r\nCannot change attribute...\r\n");
      textcolor(YELLOW); goto l10;
    }
    if (attrib & _A_RDONLY)
    { cprintf("\r\n\r\nThe file is Read Only file"
              "\r\nDo you want to make it Non Read Only ? (Y/y for yes) ");
      sound1(); if (toupper(getche()) == 'Y') attrib &= ~_A_RDONLY;
    } else
    { cprintf("\r\n\r\nThe file is Non Read Only file"
              "\r\nDo you want to make it Read Only ? (Y/y for yes) "); sound1();
      if (toupper(getche()) == 'Y') attrib |= _A_RDONLY;
    }
```

```
    if (attrib & _A_HIDDEN)
    { cprintf("\r\n\r\nThe file is Hidden file"
              "\r\nDo you want to make it Non Hidden ? (Y/y for yes) "); sound1();
      if (toupper(getche()) == 'Y') attrib &= ~_A_HIDDEN;
    } else
    { cprintf("\r\n\r\nThe file is Non Hidden file"
              "\r\nDo you want to make it Hidden ? (Y/y for yes) "); sound1();
      if (toupper(getche()) == 'Y') attrib |= _A_HIDDEN;
    }
    if (attrib & _A_SYSTEM)
    { cprintf("\r\n\r\nThe file is System file"
              "\r\nDo you want to make it Non System file ? (Y/y for yes) ");
      sound1(); if (toupper(getche()) == 'Y') attrib &= ~_A_SYSTEM;
    } else
    { cprintf("\r\n\r\nThe file is Non System file"
              "\r\nDo you want to make it System file ? (Y/y for yes) "); sound1();
      if (toupper(getche()) == 'Y') attrib |= _A_SYSTEM;
    }
    if (attrib & _A_ARCH)
    { cprintf("\r\n\r\nThe file is Archive file"
              "\r\nDo you want to make it Non Archive ? (Y/y for yes) "); sound1();
      if (toupper(getche()) == 'Y') attrib &= ~_A_ARCH;
    } else
    { cprintf("\r\n\r\nThe file is Non Archive file"
              "\r\nDo you want to make it Archive ? (Y/y for yes) "); sound1();
      if (toupper(getche()) == 'Y') attrib |= _A_ARCH;
    }
l10:if (_dos_setfileattr(file_name, attrib) != 0)
    cprintf("\r\nUnable to set file attributes"); else
    cprintf("\r\n\r\nThe attribute(s) of file...%s...are set as\r\n", file_name);
    findfirst(file_name, &ffblk, -1); display_file_attributes();
    return(0);
  } else cprintf("\r\n\r\nAttributes are not disturbed...\r\n");
  return(0);
} //Set attributes
search_files()
{ FILE *filevar;
  int count, count1, count2, linenumber, found;
  char oneline[255], substring[255], oneline_old[255], substring_old[255];
  get_check_file_name2(); count = count1 = count2 = 0;
  cprintf("Search for(Maximum 255 characters) ? (Hit just Enter to quit) ");
  sound1(); scanf("%[^\n]", substring);
  if (strlen(substring) == 0) { cprintf("\r\n"); goto l10; }
  strcpy(substring_old, substring); textcolor(CYAN);
  cprintf("\r\nDo you want to ignore case ? (Y/y for yes) ");
  sound1(); ignore_case = toupper(getche());
  cprintf("\r\n\r\n"); textcolor(YELLOW);
  if (ignore_case == 'Y')
  for (i = 0; i <= strlen(substring) - 1; i++)
  substring[i] = toupper(substring[i]);
  cprintf("Searching...%s\r\nfor...%s\r\n", file_name, substring_old);
  fnsplit(file_name, dn1, d1, n1, e1); done = findfirst(file_name, &ffblk, -1);
  found = FALSE;
  for (i = 0; i <= 254; i++) { oneline[i] = '\0'; oneline_old[i] = '\0'; }
```

```
  while (!done)
  { count++;
    cprintf("Searching file number...%5d, Name = %12s, Size = %10ld Bytes\r\n",
            count, ffblk.ff_name, ffblk.ff_fsize);
    strcpy(f_name, dn1); strcat(f_name, d1); strcat(f_name, ffblk.ff_name);
    filevar = fopen(f_name, "rt");
    if (filevar == NULL) goto l11;
    linenumber = 0;
    while (!feof(filevar))
    { l = 0; ch = '\0';
      while ((ch != '\n') && (l <= 254))
      { ch = fgetc(filevar); if (ch == EOF) goto l13;
        oneline[l] = ch; oneline_old[l] = ch; l++;
      }
l13:  linenumber++; j = 0;
      if (strlen(oneline) == 0) goto l11;
      if (ignore_case == 'Y')
      for (i = 0; i <= strlen(oneline) - 1; i++) oneline[i] = toupper(oneline[i]);
      if (strstr(oneline, substring) != '\0')
      { cprintf("%5d:%s\r", linenumber, oneline_old);
        count1++; found = TRUE;
        k = count1 / 8;
        if (count1 == k * 8)
        { cprintf("\r\nEight lines displayed...Hit any key ");
          sound1(); getch(); cprintf("\r\n");
        }
      }
      for (i = 0; i <= 254; i++) { oneline[i] = '\0'; oneline_old[i] = '\0'; }
    }
l11:if (found)
    { textcolor(CYAN);
      cprintf("\r\nNumber of lines with the string in the file = %d\r\n"
              "\r\nEnter S/s to stop or other character to continue ", count1);
      textcolor(YELLOW); sound1(); ch = toupper(getche()); cprintf("\r\n\r\n");
      if (ch == 'S') goto l12;
      count1 = 0; count2++; found = FALSE;
    }
    fclose(filevar); done = findnext(&ffblk);
  }
l12:cprintf("\r\nNumber of files searched = %d\r\n"
            "\r\nNumber of files having the string = %d\r\n"
            "\r\nFile(s) search over...\r\n", count, count2);
l10:cprintf("Hit any key to continue "); sound1(); getch(); cprintf("\r\n");
  return(0);
} //Search files
erase_file()
{ int c_d = 0;
  fnsplit(asciiz, dn1, d1, n1, e1); textcolor(CYAN);
  strcpy(f_name, dn1); strcat(f_name, d1); strcat(f_name, ffblk.ff_name);
  if ((ffblk.ff_attrib & FA_LABEL)  != 0) { cprintf("<Volume Id>"); c_d = 1; }
  if ((ffblk.ff_attrib & FA_RDONLY) != 0) { cprintf("<Read Only>"); c_d = 1; }
  if ((ffblk.ff_attrib & FA_HIDDEN) != 0) { cprintf("<Hidden>");    c_d = 1; }
  if ((ffblk.ff_attrib & FA_SYSTEM) != 0) { cprintf("<System>");    c_d = 1; }
  if ((ffblk.ff_attrib & FA_DIREC)  != 0) { cprintf("<Directory>"); c_d = 1; }
```

```
  if (c_d == 1)
  { textcolor(YELLOW);
    cprintf(" file...\r\nCannot delete...Hit any key to continue ");
    sound1(); getch(); return(0);
  }
  if ((ffblk.ff_attrib & FA_ARCH)!=0){ remove(f_name); i++; cprintf("...Deleted"); }
  textcolor(YELLOW);
  return(0);
} //Erase file
delete_files()
{ get_check_file_name2();
  done = findfirst(asciiz, &ffblk, -1);
  if (done)
  { textcolor(CYAN); cprintf("No such file(s) exist..."); textcolor(YELLOW);
    i = 0; goto l20;
  }
  cprintf("Confirm delete\r\n%s ? (Y/y for yes) ", asciiz);
  sound1(); i = 0;
  if (toupper(getche()) != 'Y') cprintf("\r\n\r\nDeletion aborted...\r\n");
  else
  { if ((strstr(asciiz, "*") != NULL) || (strstr(asciiz, "?") != NULL))
    { textcolor(CYAN);cprintf("\r\n\r\nDelete by query ? (Y/y for yes) ");
      sound1(); textcolor(YELLOW);
      if (toupper(getche()) == 'Y') goto l10;
    }
    cprintf("\r\n"); done = findfirst(asciiz, &ffblk, -1);
    while (!done)
    { cprintf("\r\nFile - %-12s", ffblk.ff_name); erase_file();
      done = findnext(&ffblk);
    }
    goto l20;
l10:cprintf("\r\n"); done = findfirst(asciiz, &ffblk, -1);
    while (!done)
    { cprintf("\r\nDelete File - %-12s ? (Y/y for yes) ", ffblk.ff_name); sound1();
      if (toupper(getche()) == 'Y') erase_file();
      done = findnext(&ffblk);
    }
l20:cprintf("\r\n\r\nNumber of files deleted = %d\r\n", i);
  }
  textcolor(YELLOW);
  return(0);
} //Delete files
clear_screen()
{ for (j = 1; j <= 50; j++) cprintf("\r\n");
  return(0);
} //Clear screen
get_no_of_chars()
{ for (i = 133; i >= 0; i--)
  if (temp[i] == '\0') continue; else break;
  return(0);
} //Get number of characters
list_lines(int start, int stop)
{ tot_ch = 0; done = FALSE; link = head -> next;
  while ((link != NULL) && !(done))
  if (link -> line_no > stop) done = TRUE;
  else if (link -> line_no >= start)
```

```
  { if ((option != 'W') && (option != 'F'))
    cprintf("%4d:", link -> line_no);
    strncpy(temp, link -> line_text, 134);
    get_no_of_chars();
    for (i1 = 0; i1 <= i; i1++)
    { ch = temp[i1];
      if ((option != 'W') && (option != 'F') && (ch != '\n')) cprintf("%c", ch);
      if ((option != 'W') && (option != 'F') && (ch == '\n')) cprintf("\r\n");
      tot_ch++;
    }
    link = link -> next;
  } else link = link -> next;
  return(0);
} //List lines
copy_old_file()
{ tot_ch_cp = 0; from_file = fopen(file_name, "r");
  to_file = fopen(f_name, "w"); ch = '\0';
  while (ch != EOF)
  { ch = fgetc(from_file);
    if (ch != EOF) { tot_ch_cp++; fputc(ch, to_file); }
    if (ch == '\n') tot_ch_cp++;
  }
  textcolor(CYAN);
  cprintf("\r\nOriginal file contents... %ld Bytes copied in...\r\n%s\r\n",
          tot_ch_cp, f_name); textcolor(YELLOW);
  cprintf("\r\nHit any key to continue ");
  sound1(); getch(); fclose(from_file); fclose(to_file);
  return(0);
} //Copy old file
file_not_disturbed()
{ cprintf("\r\n\r\nOld contents of the file are not disturbed..."
          "Hit any key to check ");
  sound1(); getch(); cprintf("\r\n");
  return(0);
} //File not disturbed
write_lines_in_file()
{ int c_w = 0;
  list_lines(number, stop); //c_w = Cannot write
  cprintf("\r\nSpace required on disk = %ld Bytes\r\n\r\n",
           tot_ch + stop - number + 1);
  done = findfirst(file_name, &ffblk, -1); textcolor(CYAN);
  if (!done)
  { if ((ffblk.ff_attrib & FA_LABEL)  != 0) { cprintf("<Volume Id>"); c_w = 1; }
    if ((ffblk.ff_attrib & FA_RDONLY) != 0) { cprintf("<Read Only>"); c_w = 1; }
    if ((ffblk.ff_attrib & FA_HIDDEN) != 0) { cprintf("<Hidden>");    c_w = 1; }
    if ((ffblk.ff_attrib & FA_SYSTEM) != 0) { cprintf("<System>");    c_w = 1; }
    if ((ffblk.ff_attrib & FA_DIREC)  != 0) { cprintf("<Directory>"); c_w = 1; }
    if (c_w == 1)
    { textcolor(YELLOW);
      cprintf(" file...Cannot write....Hit any key to continue ");
      sound1(); getch(); cprintf("\r\n");
      return(0);
    }
  }
```

```
  textcolor(YELLOW); f = fopen(file_name, "r"); fclose(f);
  if (f == NULL)
  { f = fopen(file_name, "w"); fclose(f);
    cprintf("\r\nFile...%s\r\n"
            "...is not found...New file with size = 0 is created", file_name);
  }
  else
  { done = findfirst(file_name, &ffblk, -1); sound1();
    cprintf("File...%s, Size...%ld Bytes\r\n"
            "Already exists...Overwrite it(old contents will be replaced) ? "
            "(Y/y for yes) ", file_name, ffblk.ff_fsize);
    flushall();
    if (toupper(getche()) != 'Y') { file_not_disturbed(); return(0); }
    cprintf("\r\n"); strncpy(old_file_name, file_name, 80);
    fnsplit(file_name, dn1, d1, n1, e1);
    strcpy(file_name,dn1); strcat(file_name,d1);
    strcat(file_name, n1); strcat(file_name,".OLD");
    strncpy(f_name, file_name, 80);
    strncpy(file_name, old_file_name, 80); copy_old_file();
  }
  cprintf("\r\n\r\n"); dis_file_names();
  cprintf("\r\n\r\nDo you surely want to write ? (Y/y for yes) ");
  flushall(); sound1();
  if (toupper(getche()) != 'Y') { file_not_disturbed(); return(0); }
  f = fopen(file_name, "w"); cprintf("\r\n");
  if ((f != 0) && (tot_ch > 0))
  { tot_ch = 0; done = FALSE; link = head -> next; start = number;
    while ((link != NULL) && !(done))
    if (link -> line_no > stop) done = TRUE;
    else if (link -> line_no >= start)
    { strncpy(temp, link -> line_text, 134); get_no_of_chars();
      for (nu = 0; nu <= i; nu++) { fputc(temp[nu], f); tot_ch++; }
      link = link -> next;
    } else link = link -> next;
    fclose(f); textcolor(CYAN);
    cprintf("\r\nTotal lines written = %d, Total characters written = %ld\r\n\r\n",
            stop - start + 1, tot_ch + stop - start + 1);
    textcolor(YELLOW);sound1(); cprintf("Writing over...Hit any key to check ");
    getch(); cprintf("\r\n");
  }
  else cprintf("\r\nTotal characters = %ld, Not written\r\n",
                tot_ch + stop – start + 1);
  return(0);
} //Write lines in file
insert_line(int number, char new_line[134])
{ available_memory = TRUE; //Insert a line in main linked list
  if ((unsigned long)coreleft() < 268)
  { textcolor(CYAN);
    cprintf("\r\nInsufficient memory to insert line in main linked list...\r\n"
            "\r\nHit any key to continue "); sound1(); getch();
    textcolor(YELLOW); available_memory = FALSE;
    return(0);
  }
  done = FALSE; prev = head; link = head -> next;
```

```
  while ((link != NULL) && !(done))
  if (link -> line_no == number)
  { p = (node *)malloc(sizeof(node)); p -> line_no = number;
    strncpy(p -> line_text, new_line, 134);
    p -> next = link; prev -> next = p;
    do
    { nu = link -> line_no; nu++;
      link -> line_no = nu; link = link -> next;
    } while (link != NULL);
    done = TRUE;
  } else if (link -> line_no > number)
  { p = (node *)malloc(sizeof(node)); p -> line_no = number;
    strncpy(p -> line_text, new_line, 134); p -> next = link;
    prev -> next = p; done = TRUE;
  } else { prev = link; link = link -> next; }
  if (!done)
  { p = (node *)malloc(sizeof(node)); p -> line_no = number;
    strncpy(p -> line_text, new_line, 134);
    p -> next = NULL; prev -> next = p;
  }
  return(0);
} //Insert a line
insert_line1(int number, char new_line1[134])
{ available_memory = TRUE; //Insert line in second linked list for copy/move
  if ((unsigned long)coreleft() < 268)
  { textcolor(CYAN);
    cprintf("\r\nInsufficient memory to insert line in second linked list...\r\n"
            "\r\nHit any key to continue ");
    sound1(); getch(); textcolor(YELLOW); available_memory = FALSE;
    return(0);
  }
  done = FALSE; prev1 = head1; link1 = head1 -> next1;
  while ((link1 != NULL) && !(done))
  if (link1 -> line_no1 == number)
  { p1 = (node1 *)malloc(sizeof(node1)); p1 -> line_no1 = number;
    strncpy(p1 -> line_text1, new_line1, 134); p1 -> next1 = link1;
    prev1 -> next1 = p1;
    do
    { nu = link1 -> line_no1; nu++; link1 -> line_no1 = nu;
      link1 = link1 -> next1;
    } while (link1 != NULL);
    done = TRUE;
  } else if (link1 -> line_no1 > number)
  { p1 = (node1 *)malloc(sizeof(node1)); p1 -> line_no1 = number;
    strncpy(p1 -> line_text1, new_line1, 134);
    p1 -> next1 = link1; prev1 -> next1 = p1; done = TRUE;
  } else { prev1 = link1; link1 = link1 -> next1; }
  if (!done)
  { p1 = (node1 *)malloc(sizeof(node1)); p1 -> line_no1 = number;
    strncpy(p1 -> line_text1, new_line1, 134); p1 -> next1 = NULL;
    prev1 -> next1 = p1;
  }
  return(0);
} //Insert a line in second linked list
```

```
read_lines_from_file(int sno) //Read from external file
{ char temp132;
  available_memory = TRUE;
  if (ffblk.ff_fsize != 0)
  { f = fopen(file_name, "r");
    nu = -1; tot_ch = 0;
    for (i = 0; i <= 133; i++) temp1[i] = '\0';
    while (!feof(f))
    { ch = fgetc(f); if (ch == EOF) goto l5;
      nu++; tot_ch++; temp[nu] = ch;
      if ((temp[nu] == '\n') || (nu == 132))
      { temp1[nu] = ch; tot_ch++;
        if ((unsigned long)coreleft() < 268)
        { textcolor(CYAN);
          cprintf("\r\nInsufficient memory to load/copy file...Use other editor."
                  "\r\n\r\nHit any key key to continue ");
          sound1(); getch(); available_memory = FALSE;
          textcolor(YELLOW); cprintf("\r\n"); fclose(f);
          return(0);
        }
        if ((nu == 132) && (temp[132] != '\n'))
        { temp132 = temp[132]; temp1[132] = '\n';
          strncpy(new_line, temp1, 134); insert_line(sno, new_line); sno++;
          for (k = 1; k <= 133; k++) temp1[k] = '\0';
          temp1[0] = temp132; tot_ch++; nu = 0; highest_no++;
        }
        else
        { strncpy(new_line, temp1, 134); insert_line(sno, new_line); sno++;
          highest_no++; nu = -1; for (k = 0; k <= 133; k++) temp1[k] = '\0';
        }
      }
      else temp1[nu] = temp[nu];
    }
l5: if (nu != -1)
    { temp1[nu+1] = '\n'; tot_ch += 2; strncpy(new_line, temp1, 134);
      insert_line(sno, new_line); sno++; highest_no++;
    }
    cprintf("\r\n"); fclose(f);
  }
  return(0);
} //Read lines from file
delete_line(int number)
{ prev = head; link = head -> next; done = FALSE;
  while ((link != NULL) && !(done))
  if (link -> line_no == number)
  { prev -> next = link -> next; free(link); done = TRUE; }
  else
  { prev = link; link = link -> next; }
  link = prev -> next;
  if (option == 'U')
  while (link != NULL)
  { nu = link -> line_no; nu--; link -> line_no = nu; link = link -> next; }
  return(0);
} //Delete a line
```

```
copy_lines(int start, int stop)
{ //Second linked list is used
  head1 = (node1 *)malloc(sizeof(node1)); head1 -> next1 = NULL;
  highest_no1 = 1; done = FALSE; link = head -> next;
  while ((link != NULL) && !(done))
  if (link -> line_no > stop) done = TRUE;
  else if (link -> line_no >= start)
  { strncpy(temp, link -> line_text, 134); strncpy(new_line, temp, 134);
    strncpy(new_line1, new_line, 134); insert_line1(highest_no1, new_line1);
    if (!(available_memory)) goto l5; highest_no1++; link = link -> next;
  } else link = link -> next;
l5:return(0);
} //Copy lines
check_ignore_case()
{ for (j = 0; j <= l - 1; j++)
  { temp14[j] = temp[i1+j];
    if (ignore_case == 'Y')
    if ((temp14[j] >= 'a') && (temp14[j] <= 'z')) temp14[j] -= 32;
  }
  return(0);
} //Check ignore case
search_string(int start, int stop, char temp12[21])
{ line_occ = tot_occ = k1 = 0; done = FALSE; link = head -> next;
  while ((link != NULL) && !(done))
  { if (link -> line_no > stop) done = TRUE;
    else if (link -> line_no >= start)
    { strncpy(temp, link -> line_text, 134); get_no_of_chars(); i1 = 0;
      while (i1 <= i)
      { check_ignore_case(); //Put input string length characters into temp14
        if (wild == 'Y')
        for (j = 0; j <= l - 1; j++) if (temp12[j] == '?') temp14[j] = '?';
        if (strcmp(temp14, temp12) != 0) i1++;
        else { line_occ++; tot_occ++; i1 += l; }
      }
      if (line_occ > 0)
      { k1++; cprintf("%4d:", link -> line_no); j = 0; ch = '\0';
        while (ch != '\n') { cprintf("%c", temp[j]); j++; ch = temp[j]; }
        textcolor(CYAN);
        cprintf("\r\nNumber of occurrences in this line = %d\r\n", line_occ);
        textcolor(YELLOW); k = k1 / 7;
        if (k1 == k * 7)
        { cprintf("\r\nSeven lines displayed...Enter S/s to stop or "
                  "Hit Enter key to continue "); sound1(); hit_enter();
          if (toupper(ch2[0]) == 'S') { cprintf("\r\n"); goto l1; }
          clear_screen();
        }
      }
    }
    link = link -> next; line_occ = 0;
  }
l1:cprintf("\r\nTotal occurrences = %ld\r\n\r\nSearch over..."
           "Hit Enter key to continue ", tot_occ);
   sound1(); hit_enter();
   return(0);
} //Search string
```

```
replace_string(int start, int stop, char temp12[21], char temp13[21])
{ occ = tot_occ = rep_done = line_rep = line_occ = line_occ1 = 0;
  done = FALSE; over_flow = 'N'; link = head -> next;
  if (query == 'Y') cprintf("\r\n");
  while ((link != NULL) && !(done))
  if (link -> line_no > stop) done = TRUE;
  else if (link -> line_no >= start)
  { strncpy(temp, link -> line_text, 134);
    strncpy(temp2, link -> line_text, 134);
    get_no_of_chars(); i1 = 0; t_ch = i;
    while (i1 <= i)
    { check_ignore_case(); //Put input string length characters into temp14
      if (wild == 'Y')
      for (j = 0; j <= l - 1; j++)
      if (temp12[j] == '?') temp14[j] = '?';
      if (strcmp(temp14, temp12) != 0) { i1++; goto l32; }
      occ++; line_occ++; line_occ1++; tot_occ++;
      if ((t_ch + (m - l) > 131) && (m > l)) over_flow = 'Y';
      if (over_flow == 'Y') cprintf("\r\n");
      if ((query == 'Y') || (over_flow == 'Y'))
      { cprintf("%4d:", link -> line_no); nu = 0;
        do { cprintf("%c", temp2[nu]); nu++; }
        while (!((temp2[nu] == '\n') || (nu == 132)));
        cprintf("\r\n");
        if (over_flow == 'Y')
        { textcolor(CYAN);
          cprintf("Line over flow...Last %d Character(s) will be lost...\r\n", m-l);
          textcolor(YELLOW);
        }
        cprintf("Original Line occurrence = %d,", line_occ1);
        textcolor(CYAN); cprintf(" Replace ? (Y/y for yes, A/a to abort) ");
        sound1(); hit_enter(); ch = toupper(ch2[0]); textcolor(YELLOW);
        if (ch == 'Y') goto l50; if (ch == 'A') goto l51;
        i1 += l; occ--; line_occ--; goto l32;
      }
l50:  if (l == m) for (j = 0; j <= l - 1; j++) temp2[i1+j] = temp13[j];
      else if (l < m)
      { j1 = i1 + (occ - 1) * (m - l);
        for (j = 0; j <= l - 1; j++) temp2[j1+j] = temp13[j];
        j = j1 + l;
        for (k = i + (m-l) * (line_occ-1); k >= j; k--) temp2[k+m-l] = temp2[k];
        j = i1 + (occ - 1) * (m - l);
        for (k = l; k <= m - 1; k++) { temp2[j+l] = temp13[k]; j++; }
        t_ch += (m - l); if (over_flow == 'Y') temp2[133] = '\0';
      }
      else
      { j1 = i1 - (occ - 1) * (l - m);
        for (j = 0; j <= m - 1; j++) temp2[j1+j] = temp13[j];
        j = (i1 + l) - (l - m) * occ;
        for (k = i1 + l; k <= i; k++) { temp2[j] = temp[k]; j++; }
        for (k = i; k >= i - line_occ * (l - m) + 1; k--) temp2[k] = '\0';
      }
      i1 += l; rep_done++; line_rep++; saved = FALSE;
l32: continue; }
```

```
l51: if (line_rep > 0)
     { if ((query == 'Y') || (over_flow == 'Y'))
       { cprintf("%4d:", link -> line_no); nu = 0;
         do { cprintf("%c", temp2[nu]); nu++; }
         while (!((temp2[nu] == '\n') || (nu == 132)));
         cprintf("\r\n\r\n");
       }
       for (i = 133; i >= 0; i--)
       if (temp2[i] == '\0') continue; else break;
       if (temp2[i-1] != '\n')
       { temp2[i] = '\n';
         if (i >= 132) temp2[132] = '\n';
         strncpy(link -> line_text, temp2, 134);
       } else strncpy(link -> line_text, temp2, 134);
     }
     if (((query == 'Y') || (over_flow == 'Y')) && (ch == 'A')) goto l52;
     over_flow = 'N'; occ = line_occ = line_rep = line_occ1 = 0;
     link = link -> next;
   } else link = link -> next;
l52:cprintf("\r\nTotal occurrences = %ld\r\n"
            "\r\nTotal replacements done = %ld\r\n"
            "\r\nReplacement over...Hit Enter key to continue ",
            tot_occ, rep_done);
    sound1(); hit_enter();
  return(0);
} //Replace string
upper_case(int start, int stop)
{ done = FALSE; link = head -> next;
  while ((link != NULL) && !(done))
  if (link -> line_no > stop) done = TRUE;
  else if (link -> line_no >= start)
  { strncpy(temp, link -> line_text, 134);
    get_no_of_chars();
    for (i1 = 0; i1 <= i; i1++)
    { ch = temp[i1]; if ((ch >= 'a') && (ch <= 'z')) temp[i1] = ch - 32;
    }
    strncpy(link -> line_text, temp, 134); link = link -> next;
  } else link = link -> next;
  return(0);
} //Upper case
lower_case(int start, int stop)
{ done = FALSE; link = head -> next;
  while ((link != NULL) && !(done))
  if (link -> line_no > stop) done = TRUE;
  else if (link -> line_no >= start)
  { strncpy(temp, link -> line_text, 134);
    get_no_of_chars();
    for (i1 = 0; i1 <= i; i1++)
    { ch = temp[i1]; if ((ch >= 'A') && (ch <= 'Z')) temp[i1] = ch + 32;
    }
    strncpy(link -> line_text, temp, 134); link = link -> next;
  } else link = link -> next;
  return(0);
} //Lower case
```

```
title_sentence_case(int start, int stop)
{ done = FALSE; link = head -> next;
  while ((link != NULL) && !(done))
  if (link -> line_no > stop) done = TRUE;
  else if (link -> line_no >= start)
  { strncpy(temp, link -> line_text, 134);
    get_no_of_chars(); j = i;
    temp[0] = toupper(temp[0]); old_ch = temp[0];
    for (i1 = 1; i1 <= j; i1++)
    { ch = temp[i1];
      if (choice == 'T')
      for (i = 0; i <= 31; i++)
      if (old_ch == word_separators[i]) temp[i1] = toupper(temp[i1]);
      if (choice == 'S')
      for (i = 0; i <= 9; i++)
      if (old_ch == sentence_separators[i])
      { temp[i1] = toupper(temp[i1]); sen_sep = TRUE; }
      if ((ch != ' ') && (sen_sep == TRUE))
      { temp[i1] = toupper(temp[i1]); sen_sep = FALSE; }
      old_ch = ch;
    }
    strncpy(link -> line_text, temp, 134); link = link -> next;
  } else link = link -> next;
  return(0);
} //Title/sentense case
page_heading(int pgno)
{ char buffer[136];
  _dos_getdate(&d); _dos_gettime(&t); fputc('\n', fp);
  sprintf(buffer, "File:%s ", file_name); fputs(buffer, fp);
  sprintf(buffer, "Date:%d/%d/%d ", d.day, d.month, d.year); fputs(buffer, fp);
  sprintf(buffer, "Day:%s ", day_name[d.dayofweek]); fputs(buffer, fp);
  sprintf(buffer, "Time:%d:%d:%d ", t.hour, t.minute, t.second); fputs(buffer, fp);
  sprintf(buffer, "Pg.No.:%d\r\n", pgno); fputs(buffer, fp);
  fputc('\n', fp); linenum = linenum + (page_heading_lines + 2);
  return(0);
} //Page heading
print_file()
{ cprintf("Printing file on parallel printer i.e. LPT1/PRN\r\n");
  strncpy(old_file_name, file_name, 80); get_check_file_name1();
  f = fopen(file_name, "r");
  if (f == 0)
  { textcolor(CYAN);
    cprintf("File does not exist/Wrong path so no printing\r\n\r\n");
    textcolor(YELLOW); goto l6;
  }
  fclose(f);
  do
  { textcolor(CYAN);
    cprintf("Please save your work before proceeding..."
            "Runtime error may occur...\r\nChoose...1..To print or 2..To abort ");
    flushall(); sound1(); gets(a); i = atoi(a); textcolor(YELLOW);
    if ((i < 1) || (i > 2))
    { cprintf("\r\nInvalid choice...Re-enter\r\n\r\n"); continue; }
  } while ((i != 1) && (i != 2));
```

```
  if (i == 2) { cprintf("\r\n"); goto l6; }
  do
  { cprintf("\r\nChoose...1..For 80 column printer or 2..For 132 column printer ");
    flushall(); sound1(); gets(a); j = atoi(a);
    if ((j < 1) || (j > 2))
    { textcolor(CYAN); cprintf("\r\nInvalid choice...Re-enter\r\n\r\n");
      textcolor(YELLOW); continue;
    }
  } while ((j != 1) && (j != 2));
  i = linenum = 0; f = fopen(file_name, "r"); fp = fopen("prn", "w"); ch = '\0';
  cprintf("\r\nDo you want page heading ? (Y/y for yes) ");
  sound1(); ch1 = toupper(getche()); if (ch1 == '\r') ch1 = 'N';
  if (ch1 == 'Y')
  { page_heading_chars = strlen(file_name) + 56;
    if (j == 1)
    if (page_heading_chars <= 80) page_heading_lines = 1;
    else page_heading_lines = 2; else
    if (page_heading_chars <= 132) page_heading_lines = 1;
    else page_heading_lines = 2;
    pgno = 1; page_heading(pgno);
  } else { linenum += 3; for (k = 1; k <= 3; k++) fputc('\n', fp); }
  while (!feof(f))
  { if (j == 1)
    Do { ch = fgetc(f); i++; if (ch == EOF) goto l5; fputc(ch, fp); }
    while ((ch != '\n') && (i != 81)); else
    do { ch = fgetc(f); i++; if (ch == EOF) goto l5; fputc(ch, fp); }
    while ((ch != '\n') && (i != 133)); ++linenum; i = 0;
    if (linenum == 69)
    { linenum = 0; for (k = 1; k <= 3; k++) fputc('\n', fp);
      if (ch1 == 'Y') { pgno++; page_heading(pgno); }
      else { linenum += 3; for (k = 1; k <= 3; k++) fputc('\n', fp); }
    }
  }
l5:fputc('\f', fp);
   cprintf("\r\n\r\nPrinting over...If machine hangs, press Alt & Enter, "
           "Choose close from title bar\r\n"); fclose(fp); fclose(f);
l6:cprintf("Hit any key to continue ");
   sound1(); getch(); cprintf("\r\n"); strncpy(file_name, old_file_name, 80);
  return(0);
} //Print file
get_lines()
{l5:cprintf("\r\nFrom which line number ? ");
  flushall(); sound1(); gets(a); number = atoi(a);
  number_check(number); if (number < 1) number = 1;
  cprintf("To   which line number ? ");
  flushall(); sound1(); gets(a); stop = atoi(a);
  number_check(stop); if (stop < 1) stop = 1;
  if (stop > highest_no - 1) stop = highest_no - 1;
  if ((number > stop) || (number > highest_no - 1))
  { textcolor(CYAN);
    cprintf("\r\nInvalid range of numbers...From > To or From > Highest number"
            "\r\nMaximum line number existing = %d. Retry...\r\n", highest_no - 1);
    textcolor(YELLOW); goto l5;
  }
  return(0);
} //Get lines
```

```
search_options()
{ textcolor(CYAN);
  cprintf("Do you want to ignore case ? (Y/y for yes) ");
  sound1(); hit_enter(); ignore_case = toupper(ch2[0]);
  cprintf("\r\n? can be used as a wild character "
          "e.g. s???o shall match with satao, spqro, etc."
          "Be careful in using it...Use it ? (Y/y for yes) ");
  sound1(); hit_enter(); wild = toupper(ch2[0]); textcolor(YELLOW);
  for (i = 0; i <= 20; i++)
  { temp12[i] = '\0'; temp14[i] = '\0'; }
  cprintf("\r\nWhich string ? (First 20 characters are considered..."
          "Hit just Enter to abort)\r\n"); sound1();
  fgets(stdin_str, sizeof(stdin_str), stdin);
  if (strlen(stdin_str) > 21)
  cprintf("Number of characters > 20, First 20 characters are considered\r\n");
  strncpy(temp12, stdin_str, sizeof(temp12));
  temp12[strlen(temp12) - 1] = '\0';
  l = strlen(temp12); cprintf("\r\n");
  if (l == 0) return(0);
  temp12[l] = '\0'; l = strlen(temp12);
  if (ignore_case == 'Y') //Change input string into capital
  for (i = 0; i <= l - 1; i++)
  { ch = temp12[i]; if ((ch >= 'a') && (ch <= 'z')) temp12[i] = ch - 32; }
  strncpy(b12, temp12, 21);
  return(0);
} //Search options
insert_append()
{ textcolor(CYAN);
  cprintf("You can have up to 132 characters per line. "
          "End each line with Enter key.\r\n"
          "New line begins automatically after 132 characters "
          "are entered in a line.\r\n"
          "Use Ctrl d/D to end appending/inserting. "
          "You can have maximum 3617 lines.\r\n");
  append_insert_update_message();
  cprintf("Save(W/F) your work at regular intervals "
          "to avoid the accidental loss of data.\r\n\r\n"); textcolor(YELLOW);
  l = -1; temp[0] = '\n';
  for (i = 1; i <= 133; i++) temp[i] = '\0';
l10:cprintf("%4d:", number);
  if (l == 0) cprintf("%c", temp[0]);
  for (i = 0; i <= 133; i++) temp1[i] = b[i] = new_line[i] = '\0';
  flushall(); insert_append1();
  if ((ch == 4) && (l != 0))
  { insert_line(number, new_line); if (!(available_memory)) goto l20;
    highest_no++; number++; cprintf("\r\n");
  }
  if (ch == 4) goto l20;
  insert_line(number, new_line); if (!(available_memory)) goto l20;
  strncpy(temp, temp1, 134); highest_no++; number++; cprintf("\r\n");
  if (l == 132) l = 0; else l = -1;
  goto l10;
l20:return(0);
} //Insert append
```

```
insert_append1()
{ int extended = 0;
l101: l = l++; ch = getch();
  if (ch == 4)  goto l83;    //EOT i.e. Control D
  if (ch == 13) goto l81;    //Carriage Return
  if (l > 131)  goto l82;    //Line overflow
  if ((ch == 8) && (l == 0)) //Backspace on the line's first char
  { gotoxy(1, wherey()); cprintf("%4d:", number); l = -1; goto l101; }
  if (ch == 8)               //Backspace
  { for (n = l; n <= 133; n++) temp[n-1] = temp[n];
    l -= 2; gotoxy(wherex() - 1, wherey());
    cprintf(" "); gotoxy(wherex() - 1, wherey());
    goto l101;
  }
  if (ch == 9)               //Horizontal Tab
  { for (n = 128; n >= l; n--) temp[n+5] = temp[n];
    for (i = l; i <= l + 4; i++) { temp[i] = ' '; cprintf("%c", temp[i]); }
    l += 4; goto l101;
  }
  if (!ch) extended = getch();
  if (!(extended)) goto l53; else
  { if ((extended == 59) || (extended == 77))                    //F1 or --> key
    { if (temp[l] == 10)
      { l--; gotoxy(wherex(), wherey()); extended = 0; goto l101; }
      cprintf("%c", temp[l]); extended = 0; goto l101;
    }
    if ((extended == 60) || (extended == 75) || (extended == 83)) //F2/<--/Del key
    { if (temp[l] == 10) { l--; gotoxy(wherex(),wherey()); extended = 0;goto l101; }
      for (n = l+1; n <= 133; n++) temp[n-1] = temp[n];
      l--; extended = 0; goto l101;
    }
    if ((extended == 61) || (extended == 79))                    //F3 or end key
    { if (temp[l] == 10){ l--; gotoxy(wherex(),wherey()); extended = 0; goto l101; }
      do
      { if (temp[l] != '\0') cprintf("%c", temp[l]);
      l++;
      } while ((temp[l] != '\n') && (l != 132));
      if (l == 132) { temp[132] = '\n'; temp[133] = '\0'; }
      l--; extended = 0; goto l101;
    }
  }
l53:for (n = 132; n >= l; n--) temp[n+1] = temp[n];
    temp[l] = ch; cprintf("%c", temp[l]); goto l101;
l81:for (i = 0; i <= l - 1; i++) temp1[i] = temp[i];
    temp1[l] = '\n'; strncpy(b, temp1, 134); strncpy(new_line, b, 134); goto l20;
l82:for (i = 0; i <= 131; i++) temp1[i] = temp[i];
    temp1[132] = '\n'; strncpy(b, temp1, 134); strncpy(new_line, b, 134);
    for (n = 132; n >= 0; n--) temp1[n+1] = temp1[n]; temp1[0] = ch; goto l20;
l83:if (l != 0)
    { for (i = 0; i <= l - 1; i++) temp1[i] = temp[i];
      temp1[l] = '\n'; strncpy(b, temp1, 134); strncpy(new_line, b, 134);
    }
l20:return(0);
} //Insert append1
```

```
usage()
{ textcolor(CYAN);
  cprintf("An Interactive Line Text Editor for Windows using Turbo C  "
          "By  Prof. K. J. Satao\r\n"); textcolor(YELLOW);
  cprintf("Usage : WCEDITOR<Enter> from the prompt or\r\n"
          "        Double Click on WCEDITOR Icon or\r\n"
          "        Select WCEDITOR Icon and Hit Enter key\r\n"
          "(Alt & Enter keys give full screen view, "
          "pressing again gives title bar)\r\n");
  return(0);
} //Usage
new_file()
{ saved = TRUE; clear_screen(); free(head); head = (node *)malloc(sizeof(node));
  head -> next = NULL;  highest_no = 1; tot_ch = 0; if (option != 'N') usage();
  get_check_file_name1(); f = fopen(file_name, "r");
  if (f == NULL)
  { f = fopen(file_name, "w"); textcolor(CYAN);
    cprintf("File...%s is not found\r\n"
            "New file with size = 0 is created\r\n", file_name);
    textcolor(YELLOW); cprintf("\r\nHit any key to continue ");
    sound1(); getch(); cprintf("\r\n"); dis_file_names(); dis_mem_cap(); fclose(f);
    cprintf("\r\nHit any key to continue "); sound1(); getch();
    cprintf("\r\n\r\nTotal lines = %d, Total characters = %ld\r\n",
             highest_no - 1, tot_ch);
  } else
  { fclose(f); read_lines_from_file(highest_no); textcolor(CYAN);
    if (!(available_memory)) cprintf("\r\nFile truncated...\r\n\r\n");
    cprintf("File...%s is opened for update\r\n", file_name);
    textcolor(YELLOW); cprintf("\r\nHit any key to continue ");
    sound1(); getch(); cprintf("\r\n"); dis_file_names(); dis_mem_cap();
    cprintf("\r\nHit any key to continue "); sound1(); getch();
    cprintf("\r\n\r\nTotal lines = %d, Total characters loaded = %ld\r\n",
            highest_no - 1, tot_ch);
  }
  return(0);
} // New file
insert_required_blank_lines(int number1)
{ temp[0] = '\n'; for (i = 1; i <= 133; i++) temp[i] = '\0';
  strncpy(b, temp, 134); strncpy(new_line, b, 134); l1 = highest_no;
  for (l = l1; l <= number1 - 1; l++)
  { insert_line(l1, new_line); if (!(available_memory)) return(0);
    highest_no++; list_lines(l, l);
  }
  return(0);
} //Insert required blank lines
use_w_or_f()
{ cprintf("\r\nUse W or F to save i.e. write in a file..."
          "\r\nHit any key to continue "); sound1(); getch(); cprintf("\r\n");
  return(0);
} //Use w or f
zero_lines()
{ textcolor(CYAN); cprintf("\r\n\r\nMaximum line number existing = 0\r\n");
  textcolor(YELLOW); cprintf("Hit any key to continue ");
  sound1(); getch(); cprintf("\r\n");
  return(0);
} //Zero lines
```