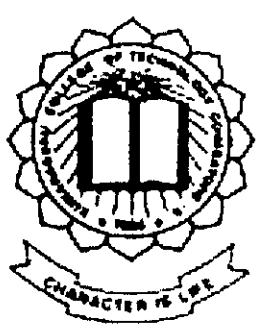
INTEGRATED DEVELOPMENT ENVIRONMENT FOR COBOL

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

OF **BHARATHIAR UNIVERSITY,**
COIMBATORE.



Submitted By

A. Praveen Tony Raja
M.Murugan
T.R.Raja

Guided By

Ms. N.Rajathi, B.E.

Department Of Computer Science and Engineering

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE - 641 006

KUMARAGURU COLLEGE OF TECHNOLOGY

Department Of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled

"INTEGRATED DEVELOPMENT ENVIRONMENT FOR COBOL"

has been submitted by

Mr. M. MURUGAN, T.R. RAJA, A. PRAVEEN TONY RAJA

in partial fulfillment of the requirements for the award of degree of
BACHELOR OF ENGINEERING in
COMPUTER SCIENCE AND ENGINEERING

branch of the Bharathiar Univeristy, Coimbatore, during the academic year 2001-2002.

N.R.Nathi
(Guide)

(Head of Department)

Submitted for the university examination held on _____

University Register Number: 9827K0719, 9827K0203
9827K0197

(Internal Examiner)

(External Examiner)

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. K.K.Padmanabhan**, Principal, Kumaraguru college of Technology, for having given us the golden opportunity to serve the purpose of our education.

We are bound to express our gratitude to **Prof.S.Thangasamy**, Head of the Department of Computer Science and Engineering, Kumaraguru College of Technology, for his constant encouragement throughout the course.

We wish to thank our Project Coordinator **Mrs.S.Devaki**, Assistant Professor, Department o Computer Science and Engineering for providing us an opportunity to prove our ability.

We wish to thank our internal guide, Senior Lecturer, **Ms.N.Rajathi**, Department of Computer Science and Engineering, Kumaraguru College of Technology, for constantly encouraging us to pursue new goals, ideas and being supportive throughout the tenure of our project.

We wish to place our unsolicited gratitude and respect to our Class Advisor, **Mrs.D.Chandrakala**, Senior Lecturer, Department of Computer Science and Engineering, Kumaraguru College of Technology.

SYNOPSIS

SYNOPSIS

The objective of the project is to design an Integrated Development Environment for COBOL which is one among the most powerful languages in scientific calculations and business applications. COBOL language was merely written using an editor and compiled and run by the operating system. But there was no way to follow the programming practices like debugging and tracing.

This condition is because for COBOL is not having an IDE. Our project is aimed at developing a handy IDE for COBOL where we can create, edit, compile and execute COBOL programs.

INDEX

INDEX

INTRODUCTION

INTRODUCTION

IMPORTANCE OF C

The increasing popularity of C is probably due to its many desirable qualities defined as follows.

- Robust language.
- C compiler combines the capabilities of Assembly language features.
- Programs written in C are efficient and fast.
- C is highly portable.
- C is well suited for structured programming.
- Ability to extend itself.

COBOL

COBOL was first published in the year 1961 as COBOL-61 by the CODASYL (Conference on DATA System Language) committee. The users started writing COBOL programs when the first COBOL compiler became available in 1965. The next revised official standard was introduced in 1974 and known as ANSI-74 COBOL or COBOL-74. The latest version of COBOL is known as COBOL-85 revised in 1985. COBOL is a strongly business and scientific calculation oriented programming language. This is a high-level language that with stood the Y2K problem.

EXISTING SYSTEMS AND ITS LIMITATIONS

Most of the recently developed languages have IDE's, but COBOL, the greatest business oriented language, doesn't have any editor available commercially for the easy working. This is because at the time when COBOL was developed, there had not been any concept of IDE.

PROPOSED SYSTEM AND ITS ADVANTAGES

We take the privilege of developing an Integrated Development Environment for COBOL which is aimed at making an easy interaction of the programmer to create, edit, compile and run COBOL programs. It is an "**All-in-One**" application for COBOL. The proposed system includes shortcut keys for all operations. The editor and the shortcut keys are similar to that of a Turbo-C editor. This property makes it easy for most users who are familiar with C programming. The time for developing a COBOL program is much reduced and quality of the program is improved while using the Integrated Development Environment.

SYSTEM RQUIREMENTS

SYSTEM REQUIREMENTS SPECIFICATION

SOFTWARE REQUIREMENTS

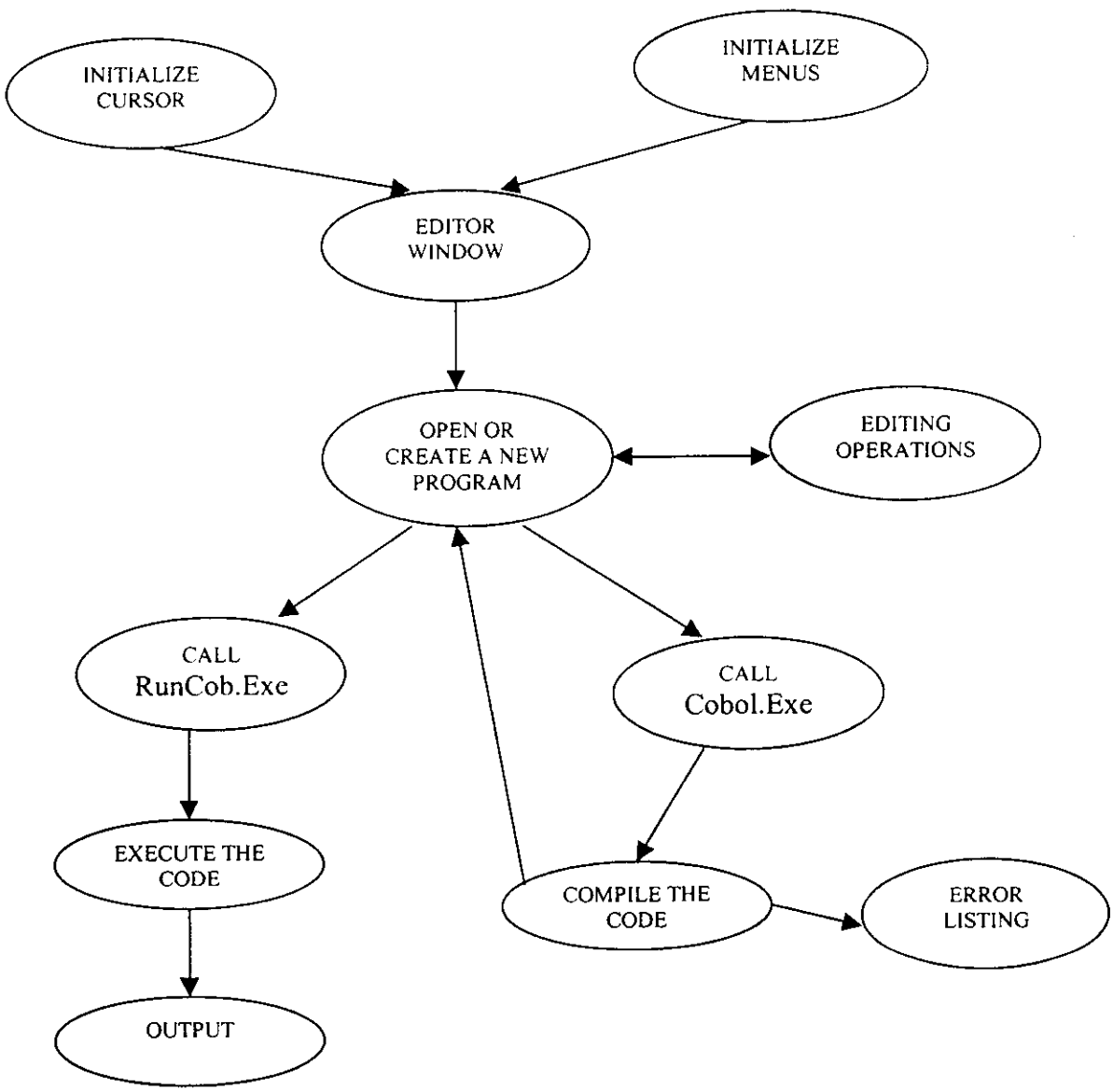
Operating System	:	DOS, Windows9X
Language	:	COBOL, Turbo-C

HARDWARE REQUIREMENTS

Processor type	:	Intel Processor
Main memory	:	8 MB RAM
Keyboard	:	Standard 100/104 Keys
Printer	:	Dot Matrix
Visual Display	:	VGA Colour Monitor

DATA FLOW DIAGRAM

DATA FLOW DIAGRAM



PRODUCT TESTING

PRODUCT TESTING

Our project of Integrated Development Environment for COBOL is tested using a COBOL program for its compilation and execution. The program is created using our application. All the editing operations such as copying a part of code, clearing a part of code or pasting the part that had already been copied to the buffer are done successfully.

The application is also tested for the proper functioning of all the shortcut keys. All the keyboard interrupts are properly handled. The **Ctrl-C interrupt** is handled to avoid the sudden termination of the program. This key code is handled by a function and made to do nothing.

Whenever a COBOL program is compiled, it is checked whether it the program is saved or not. The program is also used to check the saved state of the COBOL code whenever a new program is opened or the current program is closed on exit.

FUTURE ENHANCEMENT

FUTURE ENHANCEMENT

The current version of "**IDE for COBOL**" is able to satisfy the most common needs of the programmer for an effective creation of a COBOL program. There can be a few more new user friendly ideas that can be introduced in this project. The mouse interrupt would be a useful idea for easier working. Though the current version don't have mouse interrupt the shortcut keys does the required help for the user.

The fonts and background colour changes can be another advantage for the user, so that the user can set their own personal settings. The same can be used to take printouts in the required font and format.

CONCLUSION

CONCLUSION

The project development life cycle comes to an end with the final handover of the system. The system tries to implement most of the suggestions given by users during system study. The testing had proved the project to successful with most of the requirements being fulfilled. All the options where weighted upon and the best possible where taken into consideration and implemented.

The experience of doing this project is very awarding. Besides learning a lot of new concepts in the C-Programming language, doing this project also gave us an insight of how projects are done professionally.

REFERENCES

REFERENCES

- **C PROGRAMMING - THE ACCESSIBLE GUIDE TO PROFESSIONAL PROGRAMMING**

  By Steven Holzner with The Peter Norton Computing Group

- **ADVANCED MICROPROCESSORS & IBM - PC ALP**

  By K.UdhayaKumar and B.S.Uma Shanker

- **C PROJECTS**

  By Yashavant P.Kanetkar

- **POINTERS IN C**

  By Yashavant P.Kanetkar

- **COBOL PROGRAMMING INCLUDING MS-COBOL AND COBOL-85**

  By M.K.Roy and D.Ghosh Dastidar

APPENDIX

SOURCE CODE

```
# include "math.h"
# include "dos.h"
# include "process.h"
# include "alloc.h"
# include "stdlib.h"
# include "stdio.h"
# include "ctype.h"

# include "ed_util.c"
# include "ed_cur.c"
char far *v;
int CSD=0;
int shiftflag=0;
int strrow,strcol,endrow,endcol;
union REGS inregs, outregs ;
char *scancod, *askicod, ch, cha ;
FILE *fp1,*fp2;
char *p1;
int strrow1,strcol1,endrow1,endcol1,i, j;
int row,col;
char *p;

main ( int argc, char *argv[] )
{
        int flag ;
        /* if more than one filename is supplied at DOS prompt */
        if ( argc > 2 )
        {
                printf ( "\nInvalid number of parameters!" ) ;
                printf ( "\nPress any key..." ) ;
                fflush ( stdin ) ;
                getch() ;
                exit ( 1 ) ;
        }
        #ifdef CGA
        {
                vid_mem = ( char far * ) 0xb8000000L ;
                textmode ( 3 );                         /* Color mode = 3  */
        }
        #else
        {
```

```
        vid_mem = ( char far * ) 0xb0000000L ;
        textmode ( 7 ) ;                    /* Monochrome mode = 7 */
}
#endif
/* capture Ctrl - C interrupt */
old23 = getvect ( 0x23 ) ;
setvect ( 0x23, handler ) ;

/* capture Ctrl - Break interrupt */
old1b = getvect ( 0x1b ) ;
setvect ( 0x1b, handler ) ;

/* calculate the maximum buffer size */
/* coreleft returns a measure of RAM memory not in use.*/
maxsize = coreleft() - 5000 ;

/* allocate memory, check if successful */
buf = malloc ( maxsize ) ;
if ( buf == NULL )
        error_exit() ;

/* initialise pointers to point to buf */
startloc = endloc = curscr = currow = buf ;

/* set Ins key to on */
*ins |= 0x80 ;

/* set default file name as `NONAME.COB' */
strcpy ( filespec, "NONAME.COB" ) ;

workscreen() ; /* display working screen */
displaymenuh ( mainmenu, 6 ) ; /* display the main menu */

/* if file name to be edited is specified in the command line */
if ( argc == 2 )
{
        /* load specified file */
        strcpy ( filespec, argv[1] ) ;
        flag = load() ;

        /* if unsuccessful in loading file */
        if ( flag == 0 )
        {
                strcpy ( filespec, "NONAME.COB" ) ;
                write_fname() ;
        }
```

```
}

while ( 1 )
{
        gotoxy ( logc + 1, logr + 2 ) ;  /* position cursor */
        getkey() ;  /* receive key */

        /* display status of Insert key */
        if ( *ins & 0x80 )
                writestring ( "Insert", 24, 73, 47 ) ;
        else
                writestring ( "      ", 24, 73, 112 ) ;

        /* if special key has been hit */
        if ( ascii == 0 )
        {
                /* check which special key */
                switch ( scan )
                {
                        inregs.h.ah = 0;
                        int86(0x16, &inregs, &outregs ) ;

                        case 59 :  /* F1 key */

                                displayhelp ( 1 ) ;
                                break ;

                        case 60 :  /* F2 key */

                                save() ;
                                break ;

                        case 61 :  /* F3 key */

                                check_saved(); /* check if current file has been saved */
                                strcpy ( fname, filespec ) ;

                                /* get the name of the file to be loaded */
                                esc_flag = ask_name ( "Enter file name to Open", filespec ) ;
                                if ( esc_flag == ESC )
                                break ;

                                flag = load() ;  /* load file */

                                /* if unsuccessful in loading file */
                                if ( flag == 0 )
```

```
        {
                strcpy ( filespec, fname ) ;
                write_fname() ;
        }

        break ;


case 45 :  /* Alt - X */

        CSD=check_saved() ;
        clrscr();
        exit(0);
/*      if( CSD==1 )
        {
        clrscr();
        CSD=0;
        exit ( 0 ) ;
        }                       */
/*      else
        {
        exit ( 0 ) ;
        }           */
case 33 :  /* Alt - F */

        /* highlight the menu item */
        writestring ( mainmenu[0], 0, 2, 15 ) ;

        /* call file services */
        fserver() ;

        /* make highlighted item normal */
        writestring ( mainmenu[0], 0, 2, 112 ) ;

        break ;

case 18 :  /* Alt - E */

        writestring ( mainmenu[1], 0, 14, 15 ) ;
        eserver() ;
        writestring ( mainmenu[1], 0, 14, 112 ) ;
        break ;


case 31 :  /* Alt - S */
```

```
        writestring ( mainmenu[2], 0, 26, 15 ) ;
        sserver() ;
        writestring ( mainmenu[2], 0, 26, 112 ) ;
        break ;


case 19 :  /* Alt - R */

        writestring ( mainmenu[3], 0, 40, 15 ) ;
        rserver() ;
        writestring ( mainmenu[3], 0, 40, 112 ) ;
        break ;


case 46 :  /* Alt - C  */

        writestring ( mainmenu[4], 0, 51, 15 ) ;
        cserver() ;
        writestring ( mainmenu[4], 0, 51, 112 ) ;
        break ;


case 35 :  /* Alt - H */

        writestring ( mainmenu[5], 0, 66, 15 ) ;
        hserver() ;
        writestring ( mainmenu[5], 0, 66, 112 ) ;
        break ;

case 102 :  /* Ctrl - F9 */

        /* check if current file has been saved */
        check_saved() ;


        strcpy(dirname,"D:\\TC\\PROGS\\IDE\\RUNCOB.EXE ");
        strcpy(filename,filespec);
        strcat(dirname,filename);
        result=system(dirname);
        if (result == -1)
        {
                perror("Error from spawnl");
                exit(1);
        }
        getch();
```

```
                clrscr();
                workscreen() ;  /* display working screen */
                displaymenuh ( mainmenu, 6 ) ;  /* display the main menu */

                flag = load() ;  /* load file */

                break ;


        case 112 :  /* Alt - F9 */



                /* check if current file has been saved */
                check_saved() ;


                clrscr() ;
                strcpy( dirname, "D:\\TC\\PROGS\\IDE\\COBOL.EXE " ) ;
                strcpy( filename, filespec ) ;
                strcat( dirname, filename ) ;
                result = system( dirname ) ;

                if (result == -1)
                {
                        perror("Error from spawnl");
                        exit(1);
                }
                getch();
                clrscr();
                workscreen() ;  /* display working screen */
                displaymenuh ( mainmenu, 6 ) ;  /* display the main menu */

                flag = load() ;  /* load file */

                break ;

        case 68 :  /* F10 key */

                mm_server() ;
                break ;

        case 93 :  /* Shift F10 */

                about() ;
                break ;
```

```
case 75 :  /* left arrow key */

        inregs.h.ah = 2 ;
        int86(0x16, &inregs, &outregs ) ;
        if ( ( outregs.h.al & 0x02 ) || ( outregs.h.al & 0x01 ) )
        {
                if (shiftflag==0)
                  {
                        shiftflag=1;
                        strrow=curr;
                        strcol=curc;
                        writechar(curr,curc,' ',47);
                  }
                else
                  {
                        endrow=curr;
                        endcol=curc;
                        v = vid_mem + (endrow+1) * 160 + (endcol) * 2
;
                        fp1 = fopen ( "temp.txt", "w" ) ;
                        putc ( *v, fp1 ) ;
                        v++;
                        fclose ( fp1 ) ;
                        left() ;
                  }
        }
        else
        {
                shiftflag=0;
                left() ;
        }
        break ;




case 77 :  /* right arrow key */
        inregs.h.ah = 2 ;
        int86(0x16, &inregs, &outregs ) ;
        if ( ( outregs.h.al & 0x02 ) || ( outregs.h.al & 0x01 ) )
        {
```

```
                        if (shiftflag==0)
                         {
                                shiftflag=1;
                                strrow=curr;
                                strcol=curc;

                                writechar(curr,curc,' ',47);
                         }
                        else
                         {
                                endrow=curr;
                                endcol=curc;
                                v = vid_mem + (endrow+1) * 160 + (endcol) * 2
;
                               /* fp1 = fopen ( "temp.txt", "w" ) ;
                                putc ( *v, fp1 ) ;*/
                                v++;
                                fclose ( fp1 ) ;
                                right ();
                         }
                }
                else
                {
                        shiftflag=0;
                        right() ;
                }
                break ;


        case 99 :   /* Ctrl - F6   Copy   */

         copyopr();
         break;

        case 72 :  /* up arrow key */


                inregs.h.ah = 2 ;
                int86(0x16, &inregs, &outregs ) ;
                if ( ( outregs.h.al & 0x02 ) || ( outregs.h.al & 0x01 ) )
                {
                        if (shiftflag==0)
                         {

                                shiftflag=1;
```

```
                                strrow=curr;
                                strcol=curc;
                                writechar(curr,curc,' ',47);
                        }
                        else
                        {
                                endrow=curr;
                                endcol=curc;
                                v = vid_mem + (endrow+1) * 160 + (endcol) * 2
;
                                fp1 = fopen ( "temp.txt", "w" ) ;
                                putc ( *v, fp1 ) ;
                                v++;
                                fclose ( fp1 ) ;
                                up_line ( 1 ) ;
                        }
                }
                else
                {
                        shiftflag=0;
                        up_line ( 1 ) ;
                }
                break ;



        case 80 :  /* down arrow key */

                inregs.h.ah = 2 ;
                int86(0x16, &inregs, &outregs ) ;
                if ( ( outregs.h.al & 0x02 ) || ( outregs.h.al & 0x01 ) )
                {
                        if (shiftflag==0)
                        {
                                shiftflag=1;
                                strrow=curr;
                                strcol=curc;
                                writechar(curr,curc,' ',47);
                        }
                        else
                        {
                                endrow=curr;
                                endcol=curc;
                                v = vid_mem + (endrow+1) * 160 + (endcol) * 2
;
```

```
                        fp1 = fopen ( "temp.txt", "w" ) ;
                        putc ( *v, fp1 ) ;
                        v++;
                        fclose ( fp1 ) ;
                        down_line( 1 );
                    }
            }
            else
            {
                    shiftflag=0;
                    down_line( 1 );
            }
            break ;



case 73 :  /* PgUp key */

            page_up ( 1 ) ;
            break ;

case 81 :  /* PgDn key */

            page_down() ;
            break ;

case 71 :  /* Home key */

            start_line() ;
            break ;

case 79 :  /* End key */

            end_line() ;
            break ;


case 98 :  /* Ctrl - F5     Cut */

cutopr();
deleopr2();

break;

case 100 :   /* Ctrl -  F7  Paste */
```

```
            pasteopr();
              break ;

case 101 :  /* Ctrl - F8  Delete */

              deleopr2();


             break;


case 132 :  /* Ctrl - PgUp */

             start_file() ;
             break ;

case 118 :  /* Ctrl - PgDn */

             end_file() ;
             break ;

case 119 :  /* Ctrl - Home */

             top_screen() ;
             break ;

case 117 :  /* Ctrl - End */

             bottom_screen() ;
             break ;

case 115 :  /* Ctrl - left arrow */

             word_left() ;
             break ;

case 116 :  /* Ctrl - right arrow */

             word_right() ;
             break ;

case 83 :  /* Del key */

             del_char() ;
             break ;
```

```
                }
        }
        else
        {
                switch ( ascii )
                {
                        case 8 :  /* backspace key */

                                backspace() ;
                                break ;

                        case 20 :  /* Ctrl - T */

                                del_word_rt() ;
                                break ;

                        case 25 :  /* Ctrl - Y */

                                del_line() ;
                                break ;

                        case 12 :  /* Ctrl - L */

                                repeat_last() ;
                                break ;

                        default :

                                /* if the character is valid character */
                                if ( ( ascii >= 32 && ascii <= 126 ) || ascii == 13 || ascii == 9 )
                                        displaychar ( ascii ) ;
                }
        }

        if  ( ( ascii == 12 ) && ( scan == 38 ) )     /* Ctrl - L */
                repeat_last() ;  /* repeat last search operation */

    }
}

/* displays Main Menu, receives choice and branches control to appropriate function */
mm_server()
{
        int mchoice, esc_flag ;

        while ( 1 )
```

```
        {
                displaymenuh ( mainmenu, 6 ) ;
                mchoice = getresponseh ( mainmenu, "FESRCH", 6 ) ;

                switch ( mchoice )
                {
                        case 1 :
                                esc_flag = fserver() ;
                                break ;

                        case 2 :
                                esc_flag = eserver() ;
                                break ;

                        case 3 :
                                esc_flag = sserver() ;
                                break ;

                        case 4 :
                                esc_flag = rserver() ;
                                break ;

                        case 5 :
                                esc_flag = cserver() ;
                                break ;

                        case 6 :
                                esc_flag = hserver() ;
                                break ;


                        case ESC :  /* if Esc key is hit when in horizontal main menu */
                                esc_flag = ESC ;
                }

                /* if Esc key has been hit in vertical or horizontal menu */
                if ( esc_flag == ESC )
                        return ( esc_flag ) ;
        }
}

/* displays File menu, receives choice and branches control to appropriate function */
fserver()
{
        int fchoice, flag, esc_flag = 0 ;
        char fname[50] ;
```

```
fchoice = popupmenuv ( filemenu, 8, 1, 2, "NOSACPDQ", 1 ) ;

switch ( fchoice )
{
        case 1 :
                new() ; /* create a new file */
                break ;

        case 2 :

                check_saved() ; /* check if current file has been saved */
                strcpy ( fname, filespec ) ;

                /* get the name of the file to be loaded */
                esc_flag = ask_name ( "Enter file name to Open", filespec ) ;
                if ( esc_flag == ESC )
                        break ;

                flag = load() ; /* load file */

                /* if unsuccessful in loading file */
                if ( flag == 0 )
                {
                        strcpy ( filespec, fname ) ;
                        write_fname() ;
                }

                break ;

        case 3 :
                save() ; /* save current file */
                break ;

        case 4 :
                save_as() ; /* save current file under a new name */
                break ;

        case 5 :
                change_dir() ; /* change the default directory */
                break ;

        case 6 :
                print() ; /* print a file */
                break ;
```

```
            case 7 :
                    shell() ;  /* exit temporarily to DOS */
                    break ;



            case 8 :
                    /* check if current file has been saved */
                    check_saved() ;

                    /* restore interrupt vectors */
                    setvect ( 0x23, old23 ) ;
                    setvect ( 0x1b, old1b ) ;

                    /* exit permanently to DOS */
                    clrscr();
                    exit ( 0 ) ;


            case 75 :  /* left arrow key */

                    /* display Help menu */
                    writestring ( mainmenu[0], 0, 2, 112 ) ;
                    writestring ( mainmenu[5], 0, 66, 15 ) ;
                    esc_flag = hserver() ;
                    writestring ( mainmenu[5], 0, 66, 112 ) ;

                    break ;

            case 77 :  /* right arrow key */

                    /* display Edit menu */
                    writestring ( mainmenu[0], 0, 2, 112 ) ;
                    writestring ( mainmenu[1], 0, 14, 15 ) ;
                    esc_flag = eserver() ;
                    writestring ( mainmenu[1], 0, 14, 112 ) ;

                    break ;

            case ESC :
                    esc_flag = ESC ;
        }

        return ( esc_flag ) ;
}
```

```
/* displays Edit menu, receives choice and branches control to appropriate function */
eserver()
{
        int echoice, esc_flag ;

        char strrow[31], strcol[31], endrow[31], endcol[31], *temp ;
        int srow, scol, erow, ecol ;



        echoice = popupmenuv ( editmenu, 7, 1, 14, "TCPLDEB", 2 ) ;

        switch ( echoice )
        {
                case 1 :

                                cutopr();
                                deleopr2();
                                break;

                case 2 :
                                copyopr();
                                break;

                case 3 :
                                pastcopr();
                                break;

                case 4:
                                deleopr2();

                                break;

                case 5 :
                        del_line() ;  /* delete one line */
                        break ;

                case 6 :
                        del_line_rt() ;  /* delete line to right of cursor */
                        break ;

                case 7 :
                        del_line_lt() ;  /* delete line to left of cursor */
                        break ;
```

```
                case 75 :  /* left arrow key */

                        /* display File menu */
                        writestring ( mainmenu[1], 0, 14, 112 ) ;
                        writestring ( mainmenu[0], 0, 2, 15 ) ;
                        esc_flag = fserver() ;
                        writestring ( mainmenu[0], 0, 2, 112 ) ;

                        break ;

                case 77 :  /* right arrow key */

                        /* display Search menu */
                        writestring ( mainmenu[1], 0, 14, 112 ) ;
                        writestring ( mainmenu[2], 0, 26, 15 ) ;
                        esc_flag = sserver() ;
                        writestring ( mainmenu[2], 0, 26, 112 ) ;

                        break ;

                case ESC :
                        esc_flag = ESC ;
        }

        return ( esc_flag ) ;
}

/* displays Search menu, receives choice and branches control to appropriate function */
sserver()
{
        int schoice, esc_flag = 0 ;

        schoice = popupmenuv ( searchmenu, 4, 1, 26, "FRSG", 3 ) ;

        switch ( schoice )
        {
                case 1 :

                        /* set appropriate flags */
                        findflag = 1 ;
                        frflag = 0 ;

                        find() ;  /* search string */
                        break ;
```

```
            case 2 :

                    /* set appropriate flags */
                    findflag = 0 ;
                    frflag = 1 ;

                    replace() ;  /* search and replace string */
                    break ;

            case 3 :
                    repeat_last() ;  /* repeat last search operation */
                    break ;

            case 4 :
                    gotoline() ;  /* go to the specified line */
                    break ;


            case 75 :  /* left arrow key */

                    /* display Edit menu */
                    writestring ( mainmenu[2], 0, 26, 112 ) ;
                    writestring ( mainmenu[1], 0, 14, 15 ) ;
                    esc_flag = eserver() ;
                    writestring ( mainmenu[1], 0, 14, 112 ) ;

                    break ;

            case 77 :  /* right arrow key */

                    /* display Run menu */
                    writestring ( mainmenu[2], 0, 26, 112 ) ;
                    writestring ( mainmenu[3], 0, 40, 15 ) ;
                    esc_flag = rserver() ;
                    writestring ( mainmenu[3], 0, 40, 112 ) ;

                    break ;

            case ESC :
                    esc_flag = ESC ;
    }

    return ( esc_flag ) ;
}
```

```
/* displays Run menu, receives choice and branches control to appropriate function */
rserver()
{
        int rchoice, esc_flag = 0, result, flag ;
        char dirname[50], filename[50] ;

        rchoice = popupmenuv ( runmenu, 2, 1, 40, "RG", 4 ) ;

        switch ( rchoice )
        {
                case 1 :

                        strcpy(dirname,"D:\\TC\\PROGS\\IDE\\RUNCOB.EXE ");
                        strcpy(filename,filespec);
                        strcat(dirname,filename);
                        result=system(dirname);
                        if (result == -1)
                        {
                                perror("Error from spawnl");
                                exit(1);
                        }
                        getch();
                        clrscr();
                        workscreen() ;  /* display working screen */
                        displaymenuh ( mainmenu, 6 ) ;  /* display the main menu */

                        flag = load() ;  /* load file */


                        break ;

                case 2 :

                        gotoline() ;  /* go to the specified line */
                        break ;

                case 75 :  /* left arrow key */

                        /* display Search menu */
                        writestring ( mainmenu[3], 0, 40, 112 ) ;
                        writestring ( mainmenu[2], 0, 26, 15 ) ;

                        esc_flag = sserver() ;
                        writestring ( mainmenu[2], 0, 26, 112 ) ;
```

```
                    break ;

            case 77 :  /* right arrow key */

                    /* display Compile menu */
                    writestring ( mainmenu[3], 0, 40, 112 ) ;
                    writestring ( mainmenu[4], 0, 51, 15 ) ;
                    esc_flag = cserver() ;
                    writestring ( mainmenu[4], 0, 51, 112 ) ;

                    break ;

            case ESC :
                    esc_flag = ESC ;
    }

    return ( esc_flag ) ;
}

/* displays Compile menu, receives choice and branches control to appropriate function */
cserver()
{
    int cchoice, esc_flag = 0, result, flag ;
    char dirname[50], filename[50], intname[50], lstname[50] ;

    cchoice = popupmenuv ( compilemenu, 2, 1, 51, "CE", 5 ) ;

    switch ( cchoice )
    {
            case 1 :

                    clrscr() ;
                    strcpy( dirname, "D:\\TC\\PROGS\\IDE\\COBOL.EXE " ) ;
                    strcpy( filename, filespec ) ;
                    strcpy( lstname, " temp" )
                    strcat( dirname, filename ) ;
                    strcat( dirname, lstname ) ;
                    result = system( dirname ) ;

                    if (result == -1)
                    {
                            perror("Error from spawnl");
                            exit(1);
                    }
                    getch();
                    clrscr();
```

```
            workscreen() ;  /* display working screen */
            displaymenuh ( mainmenu, 6 ) ;  /* display the main menu */

            flag = load() ;  /* load file */

            break ;

case 2 :
            esc_flag = ask_name ( "Enter file name to Open", filespec ) ;
            if ( esc_flag == ESC )
                    break ;

            flag = load() ;  /* load file */

            /* if unsuccessful in loading file */
            if ( flag == 0 )
            {
                        strcpy ( filespec, fname ) ;
                        write_fname() ;
            }

            break ;


case 75 :  /* left arrow key */

           /* display Run menu */
           writestring ( mainmenu[4], 0, 51, 112 ) ;
           writestring ( mainmenu[3], 0, 40, 15 ) ;

           esc_flag = rserver() ;
           writestring ( mainmenu[3], 0, 40, 112 ) ;

           break ;

case 77 :  /* right arrow key */

           /* display Help menu */
           writestring ( mainmenu[4], 0, 51, 112 ) ;
           writestring ( mainmenu[5], 0, 66, 15 ) ;
           esc_flag = hserver() ;
           writestring ( mainmenu[5], 0, 66, 112 ) ;

           break ;

case ESC :
```

```
                esc_flag = ESC ;
        }

        return ( esc_flag ) ;
}


/* displays Help menu, receives choice and branches control to appropriate function */
hserver()
{
        int hchoice, esc_flag = 0 ;

        hchoice = popupmenuv ( helpmenu, 2, 1, 66, "CA", 6 ) ;

        switch ( hchoice )
        {
                case 1 :
                        displayhelp ( 1 ) ;  /* display help contents */
                        break ;

                case 2 :
                        about() ;  /* display product information message */
                        break ;


                case 75 :  /* left arrow key */

                        /* display Compile menu */
                        writestring ( mainmenu[5], 0, 66, 112 ) ;
                        writestring ( mainmenu[4], 0, 51, 15 ) ;

                        esc_flag = cserver() ;
                        writestring ( mainmenu[4], 0, 51, 112 ) ;

                        break ;

                case 77 :  /* right arrow key */

                        /* display File menu */
                        writestring ( mainmenu[5], 0, 66, 112 ) ;
                        writestring ( mainmenu[0], 0, 2, 15 ) ;
                        esc_flag = fserver() ;
                        writestring ( mainmenu[0], 0, 2, 112 ) ;

                        break ;
```

```
                case ESC :
                        esc_flag = ESC ;
        }

        return ( esc_flag ) ;
}

/* creates working screen */
workscreen()
{
        size ( 32, 0 ) ;  /* hide cursor */

        /* draw filled box in editing portion of screen */
        menubox ( 1, 0, 23, 79, 27, NO_SHADOW ) ;

        /* draw a box around editing portion of screen */
        drawbox ( 1, 0, 23, 79, 27 ) ;

        /* display the name of the current file i.e. "NONAME.COB" */
        write_fname() ;

        /* draw box of different color in bottommost row */
        menubox ( 24, 0, 24, 79, 112, NO_SHADOW ) ;

        /* display certain special keys and their significance */
        status_line() ;

        size ( 5, 7 ) ;             /* show cursor */
}

/* displays product information */
about()
{
        int area ;
        char *p ;

        size ( 32, 0 ) ;  /* hide cursor */

        /* allocate memory, if unsuccessful terminate execution */
        area = ( 17 - 6 + 1 ) * ( 60 - 19 + 1 ) * 2 ;
        p = malloc ( area ) ;
        if ( p == NULL )
                error_exit() ;

        /* create dialogue box */
        savevideo ( 6, 19, 17, 60, p ) ;
```

```
        menubox ( 6, 19, 17, 60, 112, 7 ) ;
        drawbox ( 6, 19, 16, 58 , 112 ) ;

        writestring ( "COBOL Editor", 8, 32, 112 ) ;
        writestring ( "Version 1.00", 10, 32, 112 ) ;
        writestring ( "Designed and Developed by", 11, 26, 112 ) ;
        writestring ( "Murugan, Tony, Raja", 12, 28, 112 ) ;

        /* display OK button */
        menubox ( 14, 36, 15, 43, 32, HALF_SHADOW ) ;
        writestring ( "OK", 14, 38, 47 ) ;

        /* continue till either Esc is hit or OK is selected */
        while ( 1 )
        {
                getkey() ;

                if ( ascii == ESC || ascii != 'O' || ascii == 'o' )
                        break ;
        }

        restorevideo ( 6, 19, 17, 60, p ) ;
        free ( p ) ;

        size ( 5, 7 ) ; /* show cursor */
}

/* writes the name of the file */
write_fname()
{
        int len ;
        char drive[2], fname[9], ext[5] ;

        size ( 32, 0 ) ; /* hide cursor */

        /* draw the enclosing box */
        drawbox ( 1, 0, 23, 79, 27 ) ;

        /* display current cursor location */
        writecol() ;
        writerow() ;

        /* find drive name */
        if ( filespec[1] == ':' )
                drive[0] = filespec[0] ;
        else
```

```
                drive[0] = getdisk() + 65 ;
        drive[1] = '\0' ;

        fnsplit ( filespec, "", "", fname, ext ) ;

        strcpy ( filename, " " ) ;
        strcat ( filename, drive ) ;
        strcat ( filename, ":" ) ;
        strcat ( filename, fname ) ;

        /* if extension exists */
        if ( ext[0] )
                strcat ( filename, ext ) ;

        strcat ( filename, " " ) ;
        strupr ( filename ) ;

        /* display file name */

        len = strlen ( filename ) ;
        writestring ( filename, 1, 39 - len / 2, 27 ) ;

        size ( 5, 7 ) ;  /* show cursor */
}

/* displays current row number */
writerow()
{
        int i ;
        char s[10] ;

        /* overwrite currently displayed row number */
        for ( i = 0 ; i <= 3 ; i++ )
                writechar ( 23, 60 + i, 205, 27 ) ;

        /* display current row number */
        itoa ( curr - 1, s, 10 ) ;
        writestring ( s, 23, 64 - strlen ( s ), 15 ) ;
        writechar ( 23, 64, ':', 15 ) ;

        /* position the cursor */
        gotoxy ( logc + 1, logr + 2 ) ;
}

/* displays current column number */
writecol()
```

```
{
        int i ;
        char s[10] ;

        /* overwrite currently displayed column number */
        for ( i = 0 ; i <= 2 ; i++ )
                writechar ( 23, 65 + i, 205, 27 ) ;

        /* display current column number */
        itoa ( curc, s, 10 ) ;
        writestring ( s, 23, 65, 15 ) ;
        writechar ( 23, 64, ':', 15 ) ;

        /* position the cursor */
        gotoxy ( logc + 1, logr + 2 ) ;
}

/* displays certain special keys and their significance */
status_line()
{
        menubox ( 24, 0, 24, 79, 112, NO_SHADOW ) ;
        writestring ( "^F^1 Help  ^F^2 Save  ^A^l^t^-^F^9 Compile  ^C^t^r^l^-^F^9 Run  ^A^l^t^-
^X Exit  ^F^1^0 Menu", 24, 1, 112 ) ;

        /* display current status of Ins key */
        if ( *ins & 0x80 )
                writestring ( "Insert", 24, 73, 47 ) ;
}

/* displays a message and collects the string entered in response */
ask_name ( char *str, char *name )
{
        int area, esc_flag, len ;
        char *p, currentdir[31] ;

        /* allocate memory, if unsuccessful terminate execution */
        area = ( 17 - 7 + 1 ) * ( 62 - 17 + 1 ) * 2 ;
        p = malloc ( area ) ;
        if ( p == NULL )
                error_exit() ;

        /* create dialogue box */
        savevideo ( 7, 17, 17, 62, p ) ;
        menubox ( 7, 17, 17, 62, 112, 7 ) ;
        drawbox ( 7, 17, 16, 60 , 112 ) ;
```

```
        len = strlen ( str ) ;
        writestring ( str, 9, 39 - len / 2, 112 ) ;

        menubox ( 11, 21, 12, 56, 32, HALF_SHADOW ) ;

        /* if directory name is to be entered, display current directory */
        if ( strcmp ( str, "Enter directory name" ) == 0 )
        {
                getcwd ( currentdir, 30 ) ;
                writestring ( currentdir, 11, 22, 47 ) ;
        }

        menubox ( 14, 27, 15, 51, 32, HALF_SHADOW ) ;
        writestring ( "Press Esc to cancel", 14, 29, 47 ) ;

        /* collect the string entered */
        esc_flag = getname ( 11, 22, name ) ;

        restorevideo ( 7, 17, 17, 62, p ) ;
        free ( p ) ;
        return ( esc_flag ) ;
}

/* collects a string from keyboard */
getname ( int row, int col, char *p )
{
        int i = 0 ;
        char str[30] ;

        size ( 5, 7 ) ;

        /* continue to collect characters until Esc or Enter key is hit */
        while ( 1 )
        {
                gotoxy ( col + i + 1, row + 1 ) ;
                getkey() ;

                if ( ascii == 27 )
                        return ( ESC ) ;

                /* if current directory name is displayed, erase it */
                if ( i == 0 )
                        menubox ( 11, 21, 12, 56, 32, HALF_SHADOW ) ;

                /* if Enter is hit or more than 30 characters have been entered */
                if ( ascii == 13 || i > 30 )
```

```
                break ;

            /* if backspace key is hit */
            if ( ascii == '\b' )
            {
                /* if at least one character has been entered */
                if ( i != 0 )
                {
                    i-- ;
                    writechar ( row, col + i, ' ', 47 ) ;
                }
            }

            /* if a valid ascii character and not a control character */
            if ( isascii ( ascii ) && ! iscntrl ( ascii ) )
            {
                str[i] = ascii ;
                writechar ( row, col + i, ascii, 47 ) ;
                i++ ;
            }
        }

        str[i] = '\0' ;  /* terminate string */
        strcpy ( p, str ) ;
        size ( 32, 0 ) ;
        return ( 0 ) ;
}

/* displays message strings passed to it */
message ( char *str1, char *str2 )
{
        int area, len ;
        char *p ;

        size ( 32, 0 ) ;

        /* allocate memory, if unsuccessful terminate execution */
        area = ( 17 - 8 + 1 ) * ( 60 - 19 + 1 ) * 2 ;
        p = malloc ( area ) ;
        if ( p == NULL )
                error_exit() ;

        /* create dialogue box */
        savevideo ( 8, 19, 16, 60, p ) ;
        menubox ( 8, 19, 16, 60, 112, 7 ) ;
        drawbox ( 8, 19, 15, 58 , 112 ) ;
```

```
        writestring ( " WARNING:", 9, 25, 112 ) ;
        writestring ( filename, 10, 25, 112 ) ;

        /* display the two strings */
        writestring ( str1, 10, 26 + strlen ( filename ), 112 ) ;
        len = strlen ( str2 ) ;
        writestring ( str2, 11, 39 - len / 2, 112 ) ;

        /* display Yes, No and Cancel buttons */
        menubox ( 13, 24, 14, 30, 32, HALF_SHADOW ) ;
        writestring ( " $Yes ", 13, 24, 32 ) ;
        menubox ( 13, 44, 14, 53, 32, HALF_SHADOW ) ;
        writestring ( " $Cancel ", 13, 44, 32 ) ;
        menubox ( 13, 34, 14, 39, 32, HALF_SHADOW ) ;
        writestring ( " $No ", 13, 34, 32 ) ;

        /* continue till Y, N,C or ESC is hit */
        while ( 1 )
        {
                getkey() ;
                ascii = toupper ( ascii ) ;
                if ( ascii == 'Y' || ascii == 'N' || ascii == 'C'|| ascii==ESC )
                        break ;
        }

        restorevideo ( 8, 19, 16, 60, p ) ;
        free ( p ) ;

        size ( 5, 7 ) ;
        return ( ascii ) ;
}

/* loads the specified file in memory */
load()
{
        FILE *fp ;
        int i = 0, flag = 0 ;
        char ans = 'N', *temp ;

        temp = endloc ;
        saved = YES ;
        menubox ( 24, 0, 24, 79, 112, NO_SHADOW ) ;
        write_fname() ; /* write the name of the file */
        writestring ( "Loading Editor File...", 24, 1, 112 ) ;

        /* initialise endloc so that it points to the beginning of buffer */
```

```
endloc = buf ;

/* open the specified file */
fp = fopen ( filespec, "r" ) ;

/* if unable to open file */
if ( fp == NULL )
{
        menubox ( 24, 0, 24, 79, 112, NO_SHADOW ) ;

        /* ask whether to create a new file */
        ans = message ( "does not exist...", "Create ?" ) ;
}
else
{
        /* read file contents into buffer */
        while ( ( buf[i] = getc ( fp ) ) != EOF )
        {
                i++ ;

                /* if the file size exceeds the buffer size */
                if ( i == maxsize )
                {
                        ans = message ( "too large!", "Truncate ?" ) ;

                        /* if file is to be truncated */
                        if ( ans == 'Y' )
                                break ;
                        else
                        {
                                endloc = temp ;
                                status_line() ;
                                return ( 0 ) ;
                        }
                }

                endloc++ ;
        }
}

/* if loading was successful or if new file is to be created */
if ( fp != NULL || ans == 'Y' )
{
        /* reset variables */
        curr = 2 ;
        curc = 1 ;
```

```
                logr = 1 ;
                logc = 1 ;
                skip = 0 ;
                startloc = curscr = currow = buf ;

                /* display current cursor location */
                writerow() ;
                writecol() ;

                /* clear previous screen contents */
                menubox ( 2, 1, 22, 78, 27, NO_SHADOW ) ;

                /* display one screen-full (or less) of loaded file */
                displayscreen ( curscr ) ;

                /* store the name of the file in the pick list */
                strcpy ( pickfile[pickfileno], filespec ) ;
                pickfileno++ ; /* increment the number of pick files */

                if ( pickfileno > 4 ) /* a maximum of 5 files are present in the pick list */
                        pickfileno = 0 ;
                flag = 1 ;
                status_line() ;
        }
        else
        {
                endloc = temp ;
                status_line() ;
                return ( 0 ) ;
        }

        /* close the file */
        fclose ( fp ) ;

        return ( flag ) ;
}

/* checks if current file is saved or not */
int check_saved()
{
        char ans ;
        int AAA;
        AAA=0;


        /* if file is not saved */
```

```
        if ( saved == NO )
        {
                ans = message ( "is not saved...", "Save ?" ) ;

                if ( ans == 'Y' )
                {
                        save() ;
                        AAA=1;

                }
                if ( ans== ESC)
                {
                        AAA=0;
                }

                return AAA;
        }
}

/* displays a line and returns 0 if end of file is encountered while printing that line and returns 1
otherwise */
displayline ( char *p, int row )
{
        int col, tabflag = 0, i, num ;

        if ( p >= endloc )
                return ( 0 ) ;

        num = skip ;

        /* skip past `skip' number of characters at the beginning of the line */
        for ( i = 1 ; i <= skip ; i++ )
        {
                /* if a newline is encountered */
                if ( *p == '\n' )
                        return ( 1 ) ;

                /* if a tab is encountered */
                if ( *p == '\t' )
                {
                        /* if less than 8 characters remain to be skipped */
                        if ( num <= 8 )
                                tabflag = 1 ;
                        else
                        {
                                /* skip past the tab */
```

```
                        i += 7 ;
                        num -= 8 ;

                        p++ ;
                        if ( p >= endloc )
                                return ( 0 ) ;
                }
        }
        else
        {
                p++ ;
                if ( p >= endloc )
                        return ( 0 ) ;
        }
}

/* display the line */
for ( col = 1 ; col < 79 ; col++ )
{
        if ( *p == '\n' )
                return ( 1 ) ;

        if ( *p == '\t' )
        {
                if ( tabflag )
                {
                        col += ( 7 - num ) ;  /* leave spaces representing part of the tab not
scrolled past horizontally */
                        tabflag = 0 ;
                }
                else
                        col += 7 ;
        }
        else
                writechar ( row, col, *p, 27 ) ;

        p++ ;
        if ( p >= endloc )
                return ( 0 ) ;
}

return ( 1 ) ;
}

/* displays one screen full (or less) of file contents on screen */
displayscreen ( char *p )
```

```
{
        int row, status ;

        for ( row = 2 ; row < 23 ; row++ )
        {
                /* print one line */
                status = displayline ( p, row ) ;

                /* if end of file is reached while printing the line */
                if ( status == 0 )
                        return ( 0 ) ;

                /* increment the pointer to point to the beginning of next line */
                while ( *p != '\n' )
                {
                        p++ ;

                        /* if p reaches beyond the last character in the file */
                        if ( p >= endloc )
                                return ( 0 ) ;
                }
                p++ ;

                /* if p reaches beyond the last character in the file */
                if ( p >= endloc )
                        return ( 0 ) ;
        }
}

/* loads selected file from the pick list */
pick()
{
        int choice, flag ;
        char fname[31] ;

        /* if pick list is empty */
        if ( pickfileno == 0 )
                return ;

        strcpy ( fname, filespec ) ;

        /* pop up pick file list */
        choice = popupmenuv ( pickfile, pickfileno, 1, 23, "", 7 ) ;

        /* if file is selected from the popped pick list */
        if ( choice != ESC )
```

```
        {
                /* check if current file has been saved */
                check_saved() ;
                strcpy ( filespec, pickfile[choice - 1] ) ;

                /* load file into buffer */
                flag = load() ;

                /* if unable to load file */
                if ( flag == 0 )
                {
                        strcpy ( filespec, fname ) ;
                        write_fname() ;
                }
        }
}

/* sets up a new file for editing */
new()
{
        /* check if current file has been saved */
        check_saved() ;

        /* set up `NONAME.COB' as the default file name */
        strcpy ( filespec, "NONAME.COB" ) ;
        write_fname() ;

        /* reset variables */
        curr = 2 ;
        curc = 1 ;
        logr = 1 ;
        logc = 1 ;
        saved = YES ;

        /* initialise pointers so that they point to the beginning of buffer */
        startloc = endloc = curscr = currow = buf ;

        /* clear previous screen contents */
        menubox ( 2, 1, 22, 78, 27, NO_SHADOW ) ;

        /* display current cursor location */
        writecol() ;
        writerow() ;
}

/* stores a file on disk */
```

```
save()
{
        FILE *fp ;
        char *p ;

        size ( 32, 0 ) ;

        /* if current file name is `NONAME.COB' */
        if ( strcmp ( filespec, "NONAME.COB" ) == 0 )
        {
                /* ask for the new file name */
                ask_name ( "Enter file name to Save", filespec ) ;

                /* write new file name */
                write_fname() ;

                /* add new file name to pick list */
                strcpy ( pickfile[pickfileno], filespec ) ;
                pickfileno++ ;
                if ( pickfileno > 4 )
                        pickfileno = 0 ;
        }

        /* open file for writing and check if successful */
        fp = fopen ( filespec, "w" ) ;
        if ( fp == NULL )
        {
                message ( "File creation error", "Return ?" ) ;
                return ( 0 ) ;
        }

        menubox ( 24, 0, 24, 79, 112, NO_SHADOW ) ;
        writestring ( "Saving Editor File...", 24, 1, 112 ) ;

        p = startloc ;

        /* write each character in the buffer into file */
        while ( p != endloc )
        {
                putc ( *p, fp ) ;
                p++ ;
        }

        fclose ( fp ) ;
        saved = YES ;
        status_line() ;  /* display status line */
```

```
        size ( 5, 7 ) ;
        return ( 1 ) ;
}

/* saves the curent file under a new name */
save_as()
{
        int success ;

        size ( 32, 0 ) ;
        success = save() ;  /* save the file under new name */

        if ( success )
        {
                /* display new file name */
                write_fname() ;

                /* update pick list */
                strcpy ( pickfile[pickfileno], filespec ) ;
                pickfileno++ ;
                if ( pickfileno > 4 )
                        pickfileno = 0 ;
        }

        size ( 5, 7 ) ;
}

/* merges another file into current file at current cursor location */
merge()
{
        int col, i ;
        unsigned count = 0 ;
        unsigned long totalsize ;
        FILE *fp ;
        char ans, str[17], *temp ;

        size ( 32, 0 ) ;

        strcpy ( str, filename ) ;

        /* receive name of file to merge */
        ask_name ( "Enter file name", filename ) ;

        /* open file and check if successful in opening */
        fp = fopen ( filename, "r" ) ;
        if ( fp == NULL )
```

```
{
        message ( "does not exist...", "OK ?" ) ;
        strcpy ( filename, str ) ;
        return ;
}

/* count characters in file to be merged */
while ( getc ( fp ) != EOF )
        count++ ;

totalsize = ( unsigned ) ( endloc - startloc ) ;
totalsize += count ;

/* check if file size exceeds the buffer size on merging */
if ( totalsize >= maxsize )
{
        ans = message ( "too large!", "Truncate ?" ) ;

        /* if file is to be truncated */
        if ( ans == 'Y' )
                count = maxsize - ( unsigned ) ( endloc - startloc ) ;
        else
                return ;
}

/* increment `temp' to point to character at current cursor location */
temp = currow ;
for ( col = 1 ; col < curc ; col++ )
{
        if ( *temp == '\t' )
                col += 7 ;

        if ( *temp == '\n' || temp == endloc )
                break ;

        temp++ ;
}

/* move characters after `temp' ahead by `count' bytes */
memmove ( temp + count , temp, endloc - temp ) ;

/* update ending location pointer */
endloc += count ;

saved = NO ;
```

```
        /* read the file to be merged into the buffer */
        rewind ( fp ) ;
        for ( i = 0 ; i < count ; i++ )
        {
                *temp = getc ( fp ) ;
                temp++ ;
        }

        /* clear screen contents from current row onwards */
        menubox ( logr + 1, 1, 22, 78, 27, NO_SHADOW ) ;

        /* update screen contents */
        displayscreen ( curscr ) ;

        strcpy ( filename, str ) ;

        size ( 5, 7 ) ;
}

/* changes default directory */
change_dir()
{
        char dirname[31], *p ;
        int status, area, esc_flag ;

        /* collect directory name */
        esc_flag = ask_name ( "Enter directory name", dirname ) ;
        if ( esc_flag )
                return ;

        status = chdir ( dirname ) ;

        /* allocate memory, if unsuccessful terminate execution */
        area = ( 17 - 8 + 1 ) * ( 60 - 19 + 1 ) * 2 ;
        p = malloc ( area ) ;
        if ( p == NULL )
                error_exit() ;

        /* create dialogue box */
        savevideo ( 8, 19, 16, 60, p ) ;
        menubox ( 8, 19, 16, 60, 112, 7 ) ;
        drawbox ( 8, 19, 15, 58 , 112 ) ;

        menubox ( 10, 21, 11, 56, 32, HALF_SHADOW ) ;
        menubox ( 13, 21, 14, 56, 32, HALF_SHADOW ) ;
```

```
        /* check if successful in changing directory */
        if ( status == 0 )
        {
                writestring ( "Directory sucessfully changed", 10, 22, 47 ) ;
                write_fname() ;
        }
        else
                writestring ( "Error in changing directory", 10, 22, 47 ) ;

        writestring ( "Press any key...", 13, 22, 47 ) ;
        fflush ( stdin ) ;
        getch() ;

        restorevideo ( 8, 19, 16, 60, p ) ;
        free ( p ) ;
}

/* prints the file on printer */
print()
{
        int area, tm, bm, pl, i, row = 1, esc_flag, top_of_page = 1 ;
        char *p, ch, topmargin[3], botmargin[3], pagelength[3], fname[31] ;
        FILE *fs ;

        /* receive the file name */
        esc_flag = ask_name ( "Enter file name", fname ) ;
        if ( esc_flag )
                return ;

        /* allocate memory, if unsuccessful terminate execution */
        area = ( 17 - 8 + 1 ) * ( 60 - 19 + 1 ) * 2 ;
        p = malloc ( area ) ;
        if ( p == NULL )
                error_exit() ;

        /* create dialogue box */
        savevideo ( 8, 19, 16, 60, p ) ;
        menubox ( 8, 19, 15, 60, 112, 7 ) ;
        drawbox ( 8, 19, 14, 58 , 112 ) ;

        /* open file and check if successful */
        fs = fopen ( fname, "r" ) ;
        if ( fs == NULL )
        {
                writestring ( "Unable to open", 10, 25, 112 ) ;
                writestring ( fname, 10, 40, 112 ) ;
```

```
                        row++ ;

                        /* if at end of page */
                        if ( row == pl - tm - bm )
                        {
                                /* skip bottom margin */
                                for ( i = 0 ; i < bm ; i++ )
                                        putc ( '\n', stdprn ) ;

                                top_of_page = 1 ;
                                row = 1 ;
                        }
                }
        }
}

/* searches a string in current file */
find()
{
        int esc_flag ;

        /* collect the string to be searched */
        esc_flag = ask_name ( "Enter search string", searchstr ) ;
        if ( esc_flag )
                return ( esc_flag ) ;

        search ( searchstr ) ;
}

/* searches string and returns a pointer to it */
char *search ( searchstr )
char *searchstr ;
{
        char *p, *temp, *t_loc ;
        int len, area, col, tr, tc, tlr, tlc ;

        /* initialise temporary variables */
        t_loc = currow ;
        tr = curr ;
        tc = curc ;
        tlr = logr ;
        tlc = logc ;

        len = strlen ( searchstr ) ;

        /* increment `temp' to point to character at current cursor location */
```

```
        writestring ( "Press any key to return...", 11, 24, 112 ) ;
        fflush ( stdin ) ;
        getch() ;
        restorevideo ( 8, 19, 16, 60, p ) ;
        free ( p ) ;
        return ;
}

/* collect page specifications */
esc_flag = ask_name ( "Top Margin", topmargin ) ;
esc_flag = ask_name ( "Bottom Margin", botmargin ) ;
esc_flag = ask_name ( "Page Length", pagelength ) ;

tm = atoi ( topmargin ) ;
bm = atoi ( botmargin ) ;
pl = atoi ( pagelength ) ;

writestring ( "Set up the printer", 9, 27, 112 ) ;
writestring ( "Press any key when ready...", 10, 25, 112 ) ;
menubox ( 12, 27, 13, 51, 32, HALF_SHADOW ) ;
writestring ( "Press Esc to cancel", 12, 29, 47 ) ;
getkey() ;
restorevideo ( 8, 19, 16, 60, p ) ;
free ( p ) ;

if ( ascii == ESC )
        return ;

/* continue printing till end of file is reached */
while ( ( ch = fgetc ( fs ) ) != EOF )
{
        /* if at top of page */
        if ( top_of_page )
        {
                /* skip top margin */
                for ( i = 0 ; i < tm ; i++ )
                        putc ( '\n', stdprn ) ;

                top_of_page = 0 ;
        }

        putc ( ch, stdprn ) ;

        /* if end of line is encountered */
        if ( ch == '\n' )
        {
```

```
temp = currow ;
for ( col = 1 ; col < curc ; col++ )
{
        if ( *temp == '\t' )
                col += 7 ;

        if ( *temp == '\n' || temp >= endloc )
                break ;

        temp++ ;
}

/* search string until end of file is reached or string is found */
while ( strncmp ( searchstr, temp, len ) != 0 )
{
        /* if end of file is reached */
        if ( temp >= endloc )
        {
                /* allocate memory, if unsuccessful terminate execution */
                area = ( 17 - 8 + 1 ) * ( 60 - 19 + 1 ) * 2 ;
                p = malloc ( area ) ;
                if ( p == NULL )
                        error_exit() ;

                /* create dialogue box */
                savevideo ( 8, 19, 16, 60, p ) ;
                menubox ( 8, 19, 16, 60, 112, 7 ) ;
                drawbox ( 8, 19, 15, 58 , 112 ) ;

                menubox ( 10, 21, 11, 56, 32, HALF_SHADOW ) ;
                menubox ( 13, 21, 14, 56, 32, HALF_SHADOW ) ;
                writestring ( "Search unsuccessful!", 10, 22, 47 ) ;
                writestring ( "Press any key...", 13, 22, 47 ) ;
                fflush ( stdin ) ;
                getch() ;

                /* reset the variables */
                currow = t_loc ;
                curr = tr ;
                curc = tc ;
                logr = tlr ;
                logc = tlc ;

                restorevideo ( 8, 19, 16, 60, p ) ;
                free ( p ) ;
                size ( 5, 7 ) ;
```

```
                return ( 0 ) ;
        }
        else
        {
                if ( *temp == '\t' )
                {
                        curc += 8 ;
                        temp++ ;
                }
                else
                {
                        if ( *temp == '\n' )
                        {
                                /* go to beginning of next row */
                                curr++ ;
                                curc = 1 ;
                                temp++ ;
                                currow = temp ;
                        }
                        else
                        {
                                curc++ ;
                                temp++ ;
                        }
                }
        }
}

logr = 1 ;

/* position cursor at the end of search string */
curc += ( len - 1 ) ;

/* if the string searched lies beyond 78th column on that line */
if ( curc > 78 )
{
        skip = curc - 78 ;
        logc = 78 ;
}
else
{
        skip = 0 ;
        logc = curc ;
}

/* display the file from the line which contains the search string */
```

```
        curscr = currow ;
        menubox ( 2, 1, 22, 78, 27, NO_SHADOW ) ;
        displayscreen ( curscr ) ;
        writecol() ;
        writerow() ;

        size ( 5, 7 ) ;
        return ( temp ) ;
}

/* searches for a string and replaces it with another string */
replace()
{
        int esc_flag ;

        /* collect string to be searched */
        esc_flag = ask_name ( "Enter search string", searchstr ) ;
        if ( esc_flag )
                return ;

        /* collect string to be substituted */
        esc_flag = ask_name ( "Replace with", replacestr ) ;
        if ( esc_flag )
                return ;

        f_and_r ( searchstr, replacestr ) ;
}

/* searches a string and replaces it with the specified string */
f_and_r ( char *searchstr, char *replacestr )
{
        int area, ls, lr, i ;
        char *p, *temp, *wherefr, ans ;

        /* search string and set up a pointer pointing to its beginning */
        wherefr = search ( searchstr ) ;

        /* if search is unsuccessful */
        if ( wherefr == 0 )
                return ( 0 ) ;

        /* allocate memory, if unsuccessful terminate execution */
        area = ( 17 - 8 + 1 ) * ( 60 - 19 + 1 ) * 2 ;
        p = malloc ( area ) ;
        if ( p == NULL )
                error_exit() ;
```

```
/* create dialogue box */
savevideo ( 8, 19, 16, 60, p ) ;
menubox ( 9, 19, 15, 60, 112, 7 ) ;
drawbox ( 9, 19, 14, 58 , 112 ) ;

menubox ( 11, 29, 12, 48, 32, HALF_SHADOW ) ;
writestring ( "Replace (Y/N)", 11, 30, 47 ) ;

size ( 5, 7 ) ;

/* alternate cursor between searched string and message till a key is hit */
while ( !kbhit() )
{
        gotoxy ( 45, 12 ) ;
        delay ( 10 ) ;
        gotoxy ( logc + 1, logr + 2 ) ;
        delay ( 10 ) ;
}

fflush ( stdin ) ;
ans = getch() ;
restorevideo ( 8, 19, 16, 60, p ) ;
free ( p ) ;

if ( ! ( ans == 'y' || ans == 'Y' ) )
        return ( 0 ) ;

saved = NO ;

ls = strlen ( searchstr ) ;
lr = strlen ( replacestr ) ;

if ( exceed_size ( ( unsigned ) ( endloc - startloc + lr - ls ) ) )
        return ( 1 ) ;

/* move the contents of the file after the search string to accomodate the replace string */
memmove ( wherefr + lr, wherefr + ls, endloc - ( wherefr + ls ) ) ;
endloc += ( lr - ls ) ;

/* substitute the search string with the replace string */
temp = wherefr ;
for ( i = 0 ; i < lr ; i++ )
{
        *temp = replacestr[i] ;
        temp++ ;
```

```
        int number, esc_flag ;

        /* collect the line number */
        esc_flag = ask_name ( "Enter line number", lineno ) ;
        if ( esc_flag )
                return ;

        number = atoi ( lineno ) ;
        currow = startloc ;
        temp = currow ;
        curr = 2 ;
        curc = 1 ;

        /* continue till the required line is reached */
        while ( curr != ( number + 1 ) )
        {
                /* if end of file is reached */
                if ( temp >= endloc )
                        break ;

                /* if end of line is reached */
                if ( *temp == '\n' )
                {
                        curr++ ;
                        temp++ ;
                        currow = temp ;
                }
                else
                        temp++ ;
        }

        /* display file contents starting from the specified line */
        skip = 0 ;
        curscr = currow ;
        menubox ( 2, 1, 22, 78, 27, NO_SHADOW ) ;
        displayscreen ( curscr ) ;

        /* display current cursor position */
        logr = 1 ;
        logc = 1 ;
        writerow() ;
        writecol() ;

        size ( 5, 7 ) ;
}
```

```
/* deletes character to the left of cursor */
backspace()
{
        char *temp ;
        int col ;

        /* if cursor is at the first character in file */
        if ( curc == 1 && curr == 2 )
                return ;

        /* increment `temp' to point to character at current cursor location */
        temp = currow ;
        for ( col = 1 ; col < curc ; col++ )
        {
                if ( *temp == '\t' )
                        col += 7 ;

                /* if cursor is beyond the end of line */
                if ( *temp == '\n' )
                {
                        left() ;
                        return ;
                }

                temp++ ;
        }

        /* if the character to the left of cursor is '\n' */
        if ( *( temp - 1 ) == '\n' )
        {
                /* position cursor in the previous line */
                up_line ( 1 ) ;

                /* position cursor at the end of the line */
                end_line() ;

                /* delete the '\n' at the end of the line */
                del_char() ;
        }
        else
        {
                /* position cursor one column to the left */
                left() ;

                /* delete the character at current cursor location */
                del_char() ;
```

```
        }

        curc += ( lr - ls ) ;

        /* if the replaced string lies beyond 78th column on that line */
        if ( curc > 78 )
        {
                skip = curc - 78 ;
                logc = 78 ;
        }
        else
        {
                skip = 0 ;
                logc = curc ;
        }

        /* display the file from the line which contains the replaced string */
        curscr = currow ;
        menubox ( 2, 1, 22, 78, 27, NO_SHADOW ) ;
        displayscreen ( curscr ) ;
        writecol() ;
}

/* continues the last search operation */
repeat_last()
{
        /* if find flag is set, search the next occurrence of the string */
        if ( findflag )
                search ( searchstr ) ;

        /* if find and replace flag is set, search and replace the next occurrence of the string */
        if ( frflag )
                f_and_r ( searchstr, replacestr ) ;
}

/* abandons search operation */
abort_find()
{
        frflag = 0 ;
        findflag = 0 ;
}

/* displays file contents from specified line onwards */
gotoline()
{
        char lineno[31], *temp ;
```

```
        }
}

/* deletes the character at current cursor position */
del_char()
{
        char *temp ;
        int col, row, count = 0 ;

        /* if cursor is at end of file */
        if ( currow >= endloc )
                return ;

        /* increment `temp' to point to character at current cursor location */
        temp = currow ;
        for ( col = 1 ; col < curc ; col++ )
        {
                if ( temp >= endloc )
                        return ;

                if ( *temp == '\t' )
                        col += 7 ;

                /* if cursor is beyond the end of line */
                if ( *temp == '\n' )
                        break ;

                temp++ ;
        }

        if ( temp >= endloc )
                return ;

        /* if cursor is at the end of the line or beyond the end of line */
        if ( *temp == '\n' )
        {
                /* count number of spaces from end of line to current cursor position */
                count = curc - col ;

                /* rearrange buffer to move the end of line to current cursor position */
                memmove ( temp + count, temp + 1, endloc - temp ) ;

                /* put spaces from last character in the line till current cursor position */
                memset ( temp, 32, count ) ;

                endloc += ( count - 1 ) ;
```

```
                saved = NO ;

                /* display the modified line */
                menubox ( logr + 1, 1, logr + 1, 78, 27, NO_SHADOW ) ;
                displayline ( currow, logr + 1 ) ;

                /* scroll the screen after current line */
                scrollup ( logr + 2, 1, 22, 78 ) ;

                /* display the line in the last row */
                temp = currow ;
                for ( row = logr + 1 ; row < 22 ; row++ )
                {
                        /* go to the beginning of next line */
                        while ( *temp != '\n' )
                        {
                                if ( temp >= endloc )
                                        return ;
                                temp++ ;
                        }
                        temp++ ;

                        if ( temp >= endloc )
                                return ;
                }
                displayline ( temp, row ) ;
        }
        else
        {
                /* rearrange buffer to delete the character */
                memmove ( temp, temp + 1, endloc - temp ) ;

                endloc-- ;
                saved = NO ;

                /* display the modified line */
                menubox ( logr + 1, 1, logr + 1, 78, 27, NO_SHADOW ) ;
                displayline ( currow, logr + 1 ) ;
        }
}

/* deletes the line in which cursor is currently present */
del_line()
{
        char *temp ;
        int count = 1, row ;
```

```
/* if cursor is at end of file */
if ( currow == endloc )
        return ( 0 ) ;

/* count number of characters in the line to be deleted */
temp = currow ;
while ( *temp != '\n' )
{
        /* if end of file is encountered */
        if ( temp >= endloc )
                break ;

        count++ ;
        temp++ ;
}

/* if the line to be deleted is the last line and there is no Enter at the end of the line */
if ( temp >= endloc )
{
        /* position `endloc' */
        endloc -= count ;

        /* erase last line */
        menubox ( logr + 1, 1, logr + 1, 78, 27, NO_SHADOW ) ;

        /* position cursor at the beginning of previous line */
        up_line ( 1 ) ;
        start_line() ;

        return ( 0 ) ;
}

temp++ ;

/* rearrange the buffer so that current line is deleted */
memmove ( currow, temp, endloc - temp ) ;

endloc -= count ;
saved = NO ;

/* scroll the screen after current line */
scrollup ( logr + 1, 1, 22, 78 ) ;

/* display the line in the last row */
temp = currow ;
```

```
        for ( row = logr + 1 ; row < 22 ; row++ )
        {
                /* go to the beginning of next line */
                while ( *temp != '\n' )
                {
                        if ( temp >= endloc )
                                return ( 0 ) ;
                        temp++ ;
                }
                temp++ ;

                if ( temp >= endloc )
                        return ( 0 ) ;
        }
        displayline ( temp, row ) ;
}

/* deletes line to the left of current cursor position */
del_line_lt()
{
        char *temp ;
        int count, col ;

        /* if cursor is at end of file */
        if ( currow >= endloc )
                return ;

        /* count the number of characters to the left of cursor */
        temp = currow ;
        count = 0 ;
        for ( col = 1 ; col < curc ; col++ )
        {
                if ( *temp == '\t' )
                        col += 7 ;

                /* if cursor is to the right of the end of current line */
                if ( *temp == '\n' )
                {
                        del_line() ;  /* delete the entire line */
                        return ;
                }

                temp++ ;
                count++ ;
        }
```

```
        temp1 = temp ;
        count = 0 ;
        while ( *temp1 != '\n' )
        {
                if ( temp1 >= endloc )
                        break ;

                temp1++ ;
                count++ ;
        }

        /* rearrange the buffer so that line to the right of cursor is deleted */
        memmove ( temp, temp1, endloc - temp1 ) ;

        endloc -= count ;
        saved = NO ;

        /* display the modified line */
        menubox ( logr + 1, 1, logr + 1, 78, 27, NO_SHADOW ) ;
        displayline ( currow, logr + 1 ) ;
}

/* deletes the word to the right of current cursor position */
del_word_rt()
{
        char *temp, *temp1 ;
        int col, row, count = 0 ;

        /* if cursor is at end of file */
        if ( currow >= endloc )
                return ;

        /* increment `temp' to point to character at current cursor location */
        temp = currow ;
        for ( col = 1 ; col < curc ; col++ )
        {
                if ( temp >= endloc )
                        return ;

                if ( *temp == '\t' )
                        col += 7 ;

                /* if cursor is beyond the end of line */
                if ( *temp == '\n' )
                        break ;
```

```
        /* rearrange the buffer so that line to the left of cursor is deleted */
        memmove ( currow, temp, endloc - temp ) ;

        endloc -= count ;
        saved = NO ;

        /* display the modified line */
        menubox ( logr + 1, 1, logr + 1, 78, 27, NO_SHADOW ) ;
        displayline ( currow, logr + 1 ) ;

        /* position cursor at the beginning of the line */
        start_line() ;
}

/* deletes line to the right of current cursor position */
del_line_rt()
{
        char *temp, *temp1 ;
        int col, count = 0 ;

        /* if cursor is at end of file */
        if ( currow >= endloc )
                return ;

        /* increment `temp' to point to character at current cursor location */
        temp = currow ;
        for ( col = 1 ; col < curc ; col++ )
        {
                if ( temp >= endloc )
                        return ;

                if ( *temp == '\t' )
                        col += 7 ;

                /* if cursor is to the right of the end of current line */
                if ( *temp == '\n' )
                        return ;

                temp++ ;
        }

        /* if cursor is at the end of line */
        if ( *temp == '\n' )
                return ;

        /* count the number of characters to the right of cursor */
```

```
        temp++ ;
}

if ( temp >= endloc )
        return ;

/* if cursor is at the end of the line or beyond the end of line */
if ( *temp == '\n' )
{
        /* count number of spaces from end of line to current cursor position */
        count = curc - col ;

        /* rearrange buffer to move the end of line to current cursor position */
        memmove ( temp + count, temp + 1, endloc - temp ) ;

        /* put spaces from last character in line till current cursor position */
        memset ( temp, 32, count ) ;

        endloc += ( count - 1 ) ;
        saved = NO ;

        /* display the modified line */
        menubox ( logr + 1, 1, logr + 1, 78, 27, NO_SHADOW ) ;
        displayline ( currow, logr + 1 ) ;

        /* scroll the screen after current line */
        scrollup ( logr + 2, 1, 22, 78 ) ;

        /* display the line in the last row */
        temp = currow ;
        for ( row = logr + 1 ; row < 22 ; row++ )
        {
                /* go to the beginning of next line */
                while ( *temp != '\n' )
                {
                        if ( temp >= endloc )
                                return ;
                        temp++ ;
                }
                temp++ ;
                if ( temp >= endloc )
                        return ;
        }
        displayline ( temp, row ) ;
}
else
```

```
    {
        temp1 = temp ;

        /* if character at current cursor position is alphanumeric */
        if ( isalnum ( *temp1 ) )
        {
            /* continue till a non-alphanumeric character is encountered */
            while ( isalnum ( *temp1 ) )
            {
                if ( temp1 == endloc )
                    break ;

                temp1++ ;
                count++ ;
            }
        }
        else
        {
            /* go to the next character */
            temp1++ ;
            count++ ;
        }

        /* skip consecutive spaces */
        while ( *temp1 == ' ' )
        {
            if ( temp1 == endloc )
                break ;

            temp1++ ;
            count++ ;
        }

        /* rearrange buffer so that word to the right of cursor is deleted */
        memmove ( temp, temp1, endloc - temp1 ) ;
        endloc -= count ;

        /* display the modified line */
        menubox ( logr + 1, 1, logr + 1, 78, 27, NO_SHADOW ) ;
        displayline ( currow, logr + 1 ) ;
    }
}

/* takes control temporarily to DOS */
shell()
{
```

```
        int area, status ;
        char *p ;

        /* allocate memory, if unsuccessful terminate execution */
        area = ( 24 - 0 + 1 ) * ( 79 - 0 + 1 ) * 2 ;
        p = malloc ( area ) ;
        if ( p == NULL )
                error_exit() ;

        /* create dialogue box */
        savevideo ( 0, 0, 24, 79, p ) ;
        menubox ( 0, 0, 24, 79, 7, NO_SHADOW ) ;
        menubox ( 8, 21, 16, 60, 127, 47 ) ;

        drawbox ( 9, 23, 14, 56, 127 ) ;

        writestring ( "Quitting temporarily to DOS", 11, 25, 127 ) ;
        writestring ( "Type EXIT to return...", 13, 25, 127 ) ;
        getch();
        status=system("Cls");
        gotoxy ( 7, 1 ) ;
        status = system ( "C:\\COMMAND.COM" ) ;

        /* if unable to load `COMMAND.COM' */
        if ( status == -1 )
        {
                writestring ( "Oops! Cannot load COMMAND.COM!", 11, 25, 127 ) ;
                writestring ( "Press any key...", 13, 25, 127 ) ;
                fflush ( stdin ) ;
                getch() ;
        }

        restorevideo ( 0, 0, 24, 79, p ) ;
        free ( p ) ;
}

/* ensures that no action takes place if Ctrl - C or Ctrl - Break is hit */
void interrupt handler()
{
        ctrl_c_flag = 1 ;
}

/* places a character on the screen */
displaychar ( char ch )
{
        char *temp ;
```

```
int col = 1, insert ;

/* if current column exceeds 249, beep */
if ( curc >= 249 )
{
        printf ( "\a" ) ;
        return ;
}

/* check the status of Ins key */
if ( *ins & 0x80 )
        insert = YES ;
else
        insert = NO ;

/* if Enter key is hit replace it with newline */
if ( ch == '\r' )
        ch = '\n' ;

/* increment `temp' to point to character at current cursor location */
temp = currow ;
for ( col = 1 ; col < curc ; col++ )
{
        /* if cursor is beyond the end of line or the end of file is reached */
        if ( *temp == '\n' || temp >= endloc )
        {
                /* if spacebar was hit */
                if ( ch == ' ' )
                {
                        /* position cursor one column to the right */
                        right() ;
                        return ;
                }

                /* if Enter key was hit */
                if ( ch == '\n' )
                        break ;

                if ( exceed_size ( ( unsigned ) ( endloc - startloc + curc - col + 1 ) ) )
                        return ;

                /* rearrange buffer to move end of line to current cursor position */
                memmove ( temp + curc - col, temp, endloc - temp ) ;

                /* put spaces from last character in line till current cursor position */
                memset ( temp, 32, curc - col ) ;
```

```
                /* position `temp' at the end of these spaces */
                temp += curc - col ;

                endloc += curc - col ;
                saved = NO ;

                /* rearrange the buffer to accomodate the character hit */
                memmove ( temp + 1, temp, endloc - temp ) ;

                /* store the character in the buffer */
                *temp = ch ;

                endloc++ ;

                /* display the character */
                writechar ( logr + 1, logc, ch, 27 ) ;

                /* position cursor one column to the right */
                if ( ch == '\t' )
                {
                        curc += 8 ;
                        logc += 8 ;

                        /* position cursor at appropriate column */
                        gotocol() ;
                }
                else
                        right() ;

                return ;
        }

        if ( *temp == '\t' )
                col += 7 ;

        temp++ ;
}

/* if Enter key is hit */
if ( ch == '\n' )
{
        /* if cursor is at or beyond the last character in the file */
        if ( temp >= endloc )
        {
                if ( exceed_size ( ( unsigned ) ( endloc - startloc + 1 ) ) )
```

```
                return ;

        /* put the character in the buffer */
        *temp = ch ;

        endloc++ ;
        saved = NO ;

        /* erase the current line */
        menubox ( logr + 1, logc, logr + 1, 78, 27, NO_SHADOW ) ;

        /* display the modified line */
        displayline ( currow, logr + 1 ) ;

        /* position cursor at the beginning of the next line */
        down_line ( 1 ) ;
        start_line() ;
        return ;
}

/* if Ins is off */
if ( insert == NO )
{
        /* position cursor at the beginning of the next line */
        down_line ( 1 ) ;
        start_line() ;

        /* position cursor on the first non-whitespace character */
        temp = currow ;
        while ( *temp == ' ' || *temp == '\t' )
        {
                if ( *temp == '\t' )
                        curc += 7 ;

                temp++ ;
                curc++ ;
        }

        /* if the first non-whitespace character is beyond the first 78 columns */
        if ( curc > 78 )
        {
                /* scroll the screen horizontally */
                logc = 78 ;
                skip = curc - 78 ;
                menubox ( 2, 1, 22, 78, 27, NO_SHADOW ) ;
                displayscreen ( curscr ) ;
```

```
                }
                else
                        logc = curc ;

                writecol() ;
                return ;
        }
}

/* if Ins is on or end of file is encountered */
if ( insert == YES || temp == endloc || *temp == '\n' )
{
        if ( exceed_size ( ( unsigned ) ( endloc - startloc + 1 ) ) )
                return ;

        /* rearrange the buffer to accomodate the character */
        memmove ( temp + 1, temp, endloc - temp ) ;

        endloc++ ;
}

/* place the character in the buffer */
*temp = ch ;

saved = NO ;

/* if Enter is hit (Ins is on) */
if ( ch == '\n' )
{
        /* remove spaces and tabs at the end of the line */
        del_whitespace() ;

        /* erase the current row */
        menubox ( logr + 1, logc, logr + 1, 78, 27, NO_SHADOW ) ;

        /* scroll down the screen below the current line */
        scrolldown ( logr + 2, 1, 22, 78 ) ;

        /* position cursor at the beginning of the next line */
        down_line ( 1 ) ;
        start_line() ;

        /* display the modified current line */
        displayline ( currow, logr + 1 ) ;
}
else
```

```
        {
                /* erase the current line */
                menubox ( logr + 1, logc, logr + 1, 78, 27, NO_SHADOW ) ;

                /* display the modified line */
                displayline ( currow, logr + 1 ) ;

                /* if tab key is hit */
                if ( ch == '\t' )
                {
                        curc += 8 ;
                        logc += 8 ;

                        /* position cursor at appropriate column */
                        gotocol() ;
                }
                else
                        right() ;  /* position cursor in the next column */
        }
}

/* displays error message and terminates execution */
error_exit()
{
        writestring ( "Memory Allocation Error! Press any key...", 22, 14, 112 ) ;
        fflush ( stdin ) ;
        getch() ;
        exit ( 2 ) ;
}

/* removes spaces and tabs at the end of the line */
del_whitespace()
{
        char *temp ;

        /* go to the end of the line */
        temp = currow ;
        while ( *temp != '\n' )
        {
                if ( temp >= endloc )
                        return ;

                temp++ ;
        }

        /* remove tabs and spaces after the end of the line */
```

```
        while ( * ( temp - 1 ) == '\t' || * ( temp - 1 ) == ' ' )
        {
                memmove ( temp - 1, temp, endloc - temp ) ;
                temp-- ;
                endloc-- ;
        }
}

/* checks whether the maximum buffer size is exceeded */
exceed_size ( unsigned int size )
{
        int area ;
        void *p ;

        if ( size >= maxsize )
        {
                /* allocate memory, if unsuccessful terminate execution */
                area = ( 14 - 11 + 1 ) * ( 64 - 15 + 1 ) * 2 ;
                p = malloc ( area ) ;
                if ( p == NULL )
                        error_exit() ;

                /* create dialogue box */
                savevideo ( 9, 15, 15, 65, p ) ;
                menubox ( 9, 15, 15, 65, 112, 7 ) ;
                drawbox ( 9, 15, 14, 63, 112 ) ;
                writestring ( "File size too large! Delete some characters!!", 11, 17, 112 ) ;
                writestring ( "Press any key...", 12, 17, 112 ) ;
                getch() ;

                restorevideo ( 9, 15, 15, 65, p ) ;
                return ( 1 ) ;
        }

        return ( 0 ) ;
}

/* Code for Paste operation */
pasteopr()
{
        int col, i ;
        unsigned count = 0 ;
        unsigned long totalsize ;
        FILE *fp ;
        char ans, str[17], *temp ;
```

```
size ( 32, 0 ) ;

strcpy ( str, filename ) ;

/* receive name of file to merge */
/* open file and check if successful in opening */
fp = fopen ( "temp.txt", "r" ) ;

/* count characters in file to be merged */
while ( getc ( fp ) != EOF )
        count++ ;

totalsize = ( unsigned ) ( endloc - startloc ) ;
totalsize += count ;

/* check if file size exceeds the buffer size on merging */
if ( totalsize >= maxsize )
{
        ans = message ( "too large!", "Truncate ?" ) ;

        /* if file is to be truncated */
        if ( ans == 'Y' )
                count = maxsize - ( unsigned ) ( endloc - startloc ) ;
        else
                return ;
}

/* increment `temp' to point to character at current cursor location */
temp = currow ;
for ( col = 1 ; col < curc ; col++ )
{
        if ( *temp == '\t' )
                col += 7 ;

        if ( *temp == '\n' || temp == endloc )
                break ;

        temp++ ;
}

/* move characters after `temp' ahead by `count' bytes */
memmove ( temp + count , temp, endloc - temp ) ;

/* update ending location pointer */
endloc += count ;
```

```
        saved = NO ;

        /* read the file to be merged into the buffer */
        rewind ( fp ) ;
        for ( i = 0 ; i < count ; i++ )
        {
                *temp = getc ( fp ) ;
                temp++ ;
        }

        /* clear screen contents from current row onwards */
        menubox ( logr + 1, 1, 22, 78, 27, NO_SHADOW ) ;

        /* update screen contents */
        displayscreen ( curscr ) ;

        strcpy ( filename, str ) ;

        size ( 5, 7 ) ;
}

/* Code for copy operation */
copyopr()
{
                                if(shiftflag==1)
                                {
                                        writechar(curr,curc,' ',47);
                                        fp1 = fopen ( "temp.txt", "w" ) ;

                                        if (strrow<=endrow)
                                        {

                                          /* Select within a single row */
                                          if (strrow==endrow)
                                          {
                                                for ( i = strrow ; i <= endrow ; i++ )
                                                {
                                                        /* Select using Right Arrow */

                                                        if (strcol<=endcol)
                                                        {
                                                                for ( j = strcol ; j <= endcol ; j++ )
                                                                {
                                                                        v = vid_mem + i * 160 + j * 2 ;
                                                                        putc ( *v, fp1 ) ;
                                                                        v++ ;
```

```
                }
            }
            /* Select using Left Arrow */
            else
            {
                for ( j = endcol ; j <= strcol ; j++ )
                {
                    v = vid_mem + i * 160 + j * 2 ;
                    putc ( *v, fp1 ) ;
                    v++ ;
                }
            }
        }
}
/* Select using Down Arrow */
else
{
    for ( i = strrow ; i <= endrow ; i++ )
    {
        if (i>strrow)
            j=1;
        else
            j=strcol;
        if (i!=endrow)
        {
            for ( j ; j<=78 ; j++ )
            {
                v = vid_mem + i * 160 + j * 2 ;
                putc ( *v, fp1 ) ;
                if (j==78)
                  putc('\n',fp1);
                v++ ;
            }
        }
        else
        {
            for ( j ; j<=endcol ; j++ )
            {
                v = vid_mem + i * 160 + j * 2 ;
                putc ( *v, fp1 ) ;
                v++ ;
            }
        }

    }
}
```

```
                }
                /* Select using Up Arrow */
                else
                {
                        for ( i = endrow ; i <= strrow ; i++ )
                        {
                                if (i>endrow)
                                        j=1;
                                else
                                        j=endcol;
                                if (i!=strrow)
                                {
                                        for ( j ; j<=78 ; j++ )
                                        {
                                                v = vid_mem + i * 160 + j * 2 ;
                                                putc ( *v, fp1 ) ;
                                                if (j==78)
                                                  putc('\n',fp1);

                                                v++ ;
                                        }
                                }
                                else
                                {
                                        for ( j ; j<=strcol ; j++ )
                                        {
                                                v = vid_mem + i * 160 + j * 2 ;
                                                putc ( *v, fp1 ) ;
                                                v++ ;
                                        }
                                }

                        }
                }
                fclose(fp1);
        }
        shiftflag=0;

                writechar(strrow,strcol,' ',27);
                writechar(curr,curc,' ',27);
}

deleteopr()
{
```

```
strrow1=strrow;
endrow1=endrow;
strcol1=strcol;
endcol1=endcol;



if(shiftflag==1)
{

        if (strrow1<=endrow1)
        {
          /* Select within a single row */
          if (strrow1==endrow1)
          {
                for ( i = strrow1 ; i <= endrow1 ; i++ )
                {
                        /* Select using Right Arrow */
                        if (strcol1<=endcol1)
                        {

                        for ( j = endcol1 ; j >= strcol1 ; j-- )
                        {

                                v = vid_mem + i * 160 + j * 2 ;
                                backspace();
                                v++ ;
                        }
                        }
                        /* Select using Left Arrow */
                        else
                        {

                        for ( j = endcol1 ; j <= strcol1 ; j++ )
                        {
                                v = vid_mem + i * 160 + j * 2 ;
                                del_char();
                                v++ ;

                        }


                        }
                }

          }
```

```
/* Select using Down Arrow */
else
{
printf("\n\n\n\n\n%d  %d",strrow1,endrow1);
printf("\n%d  %d",strcol1,endcol1);

        for ( i = endrow1 ; i >= strrow1 ; i-- )
        {
                if (i<endrow1)
                        j=1;
                else
                        j=endcol1;
                if (i!=endrow1)
                {
                        for ( j ; j<=78 ; j++ )
                        {
                                v = vid_mem + i * 160 + j * 2 ;
                                backspace();
                                v++ ;
                        }

                }
                else
                {
                        for ( j ; j>=strcol1 ; j-- )
                        {
                                v = vid_mem + i * 160 + j * 2 ;
                                backspace();
                                v++ ;
                        }
                }

        }
 }

}
/* Select using Up Arrow */
else
{
        for ( i = endrow1 ; i <= strrow1 ; i++ )
        {
                if (i>endrow1)
                        j=1;
                else
                        j=endcol1;
                if (i!=strrow1)
```

```
                                        {
                                                for ( j ; j<=78 ; j++ )
                                                {
                                                        v = vid_mem + i * 160 + j * 2 ;
                                                        del_char();
                                                        v++ ;
                                                }
                                        }
                                        else
                                        {
                                                for ( j ; j<=strcol1 ; j++ )
                                                {
                                                        v = vid_mem + i * 160 + j * 2 ;
                                                        del_char();
                                                        v++ ;
                                                }
                                        }

                                }

                        }
                fclose(fp1);
                }
                shiftflag=0;

}

cutopr()
{

                /* Copy Operations*/
                if(shiftflag==1)
                {
                        writechar(curr,curc,' ',47);
                        fp1 = fopen ( "temp.txt", "w" ) ;

                        if (strrow<=endrow)
                        {
                          /* Select within a single row */
                          if (strrow==endrow)
                          {
                                for ( i = strrow ; i <= endrow ; i++ )
                                {
                                        /* Select using Right Arrow */
                                        if (strcol<=endcol)
                                        {
```

```
                    for ( j = strcol ; j <= endcol ; j++ )
                    {
                            v = vid_mem + i * 160 + j * 2 ;
                            putc ( *v, fp1 ) ;
                            v++ ;
                    }
                    }
                    /* Select using Left Arrow */
                    else
                    {

                    for ( j = endcol ; j <= strcol ; j++ )
                    {
                            v = vid_mem + i * 160 + j * 2 ;
                            putc ( *v, fp1 ) ;
                            v++ ;
                    }


                    }
            }

}
/* Select using Down Arrow */
else
{
        for ( i = strrow ; i <= endrow ; i++ )
        {
                if (i>strrow)
                        j=1;
                else
                        j=strcol;
                if (i!=endrow)
                {
                        for ( j ; j<=78 ; j++ )
                        {
                                v = vid_mem + i * 160 + j * 2 ;
                                putc ( *v, fp1 ) ;
                                if (j==78)
                                  putc('\n',fp1);

                                v++ ;
                        }
                }
                else
                {
```

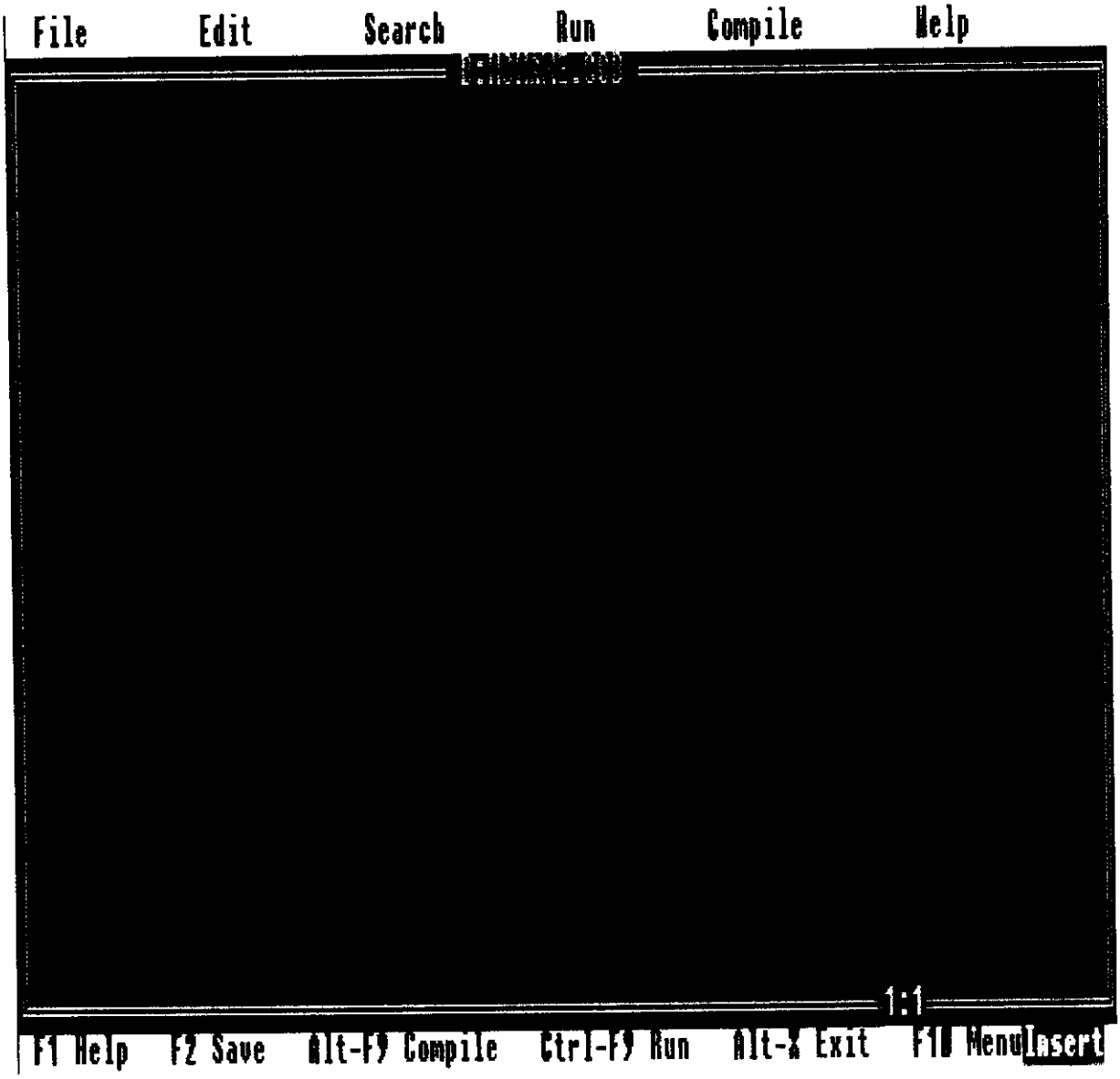
```
                                        for ( j ; j<=endcol ; j++ )
                                        {
                                                v = vid_mem + i * 160 + j * 2 ;
                                                putc ( *v, fp1 ) ;
                                                v++ ;
                                        }
                                }

                        }
                  }

                }
                /* Select using Up Arrow */
                else
                {
                        for ( i = endrow ; i <= strrow ; i++ )
                        {
                                if (i>endrow)
                                        j=1;
                                else
                                        j=endcol;
                                if (i!=strrow)
                                {
                                        for ( j ; j<=78 ; j++ )
                                        {
                                                v = vid_mem + i * 160 + j * 2 ;
                                                putc ( *v, fp1 ) ;
                                                if (j==78)
                                                  putc('\n',fp1);
                                                v++ ;
                                        }
                                }
                                else
                                {
                                        for ( j ; j<=strcol ; j++ )
                                        {
                                                v = vid_mem + i * 160 + j * 2 ;
                                                putc ( *v, fp1 ) ;
                                                v++ ;
                                        }
                                }

                        }

                }
        fclose(fp1);
```

```
                                        }
}
int mouse_init(void)
{
union REGS in,out;
in.x.ax=0;
int86(0x33,&in,&out);
return out.x.ax;
}

showcur()
{
union REGS in,out;
in.x.ax=1;
int86(0x33,&in,&out);
}

mouse_info(int *row,int *col)
{
union REGS in,out;
in.x.ax=3;
int86(0x33,&in,&out);
*row=out.x.dx/8+1;
*col=out.x.cx/8+1;
}
deleopr2()
{
                                    if(shiftflag==1)
                                    {
                                            writechar(curr,curc,' ',47);
/*                                          fp1 = fopen ( "temp.txt", "w" ) ;*/

                                            if (strrow<=endrow)
                                            {

                                              /* Select within a single row */
                                              if (strrow==endrow)
                                              {
                                                      for ( i = strrow ; i <= endrow ; i++ )
                                                      {
                                                              /* Select using Right Arrow */
                                                              if (strcol<=endcol)
                                                              {
                                                                      for ( j = strcol ; j <= endcol ; j++ )
                                                                      {
                                                                              v = vid_mem + i * 160 + j * 2 ;
```

```
                                        backspace();
                                        v++ ;
                                }
                        }
                        /* Select using Left Arrow */
                        else
                        {
                                for ( j = endcol ; j <= strcol ; j++ )
                                {
                                        v = vid_mem + i * 160 + j * 2 ;
                                        del_char();
                                        v++ ;
                                }
                        }
                }
        }
        /* Select using Down Arrow */
        else
        {
                for ( i = strrow ; i <= endrow ; i++ )
                {
                        if (i>strrow)
                                j=1;
                        else
                                j=strcol;
                        if( (i!=endrow) && (!(i<endrow)) )
                        {
                                for ( j ; j<=78 ; j++ )
                                {
                                        v = vid_mem + i * 160 + j * 2 ;
                                        if (j==78)
                                          putc('\n',fp1);
                                          backspace();
                                        v++ ;
                                }
                        }
                        else
                        {
                                for ( j ; j<=endcol ; j++ )
                                {
                                        v = vid_mem + i * 160 + j * 2 ;
                                        backspace();
                                        v++ ;
                                }
                        }
```

```
                        }
                    }

                }
                /* Select using Up Arrow */
                else
                {
                        for ( i = endrow ; i <= strrow ; i++ )
                        {
                                if (i>endrow)
                                        j=1;
                                else
                                        j=endcol;

                                if ((i!=strrow) && (!(i<strrow)))
                                {
                                        for ( j ; j<=78 ; j++ )
                                        {
                                                v = vid_mem + i * 160 + j * 2 ;
                                                putc ( *v, fp1 ) ;
                                                  putc('\n',fp1);
                                                  del_char();

                                                v++ ;
                                        }
                                }
                                else
                                {
                                        for ( j ; j<=strcol ; j++ )
                                        {
                                                v = vid_mem + i * 160 + j * 2 ;
                                                del_char();
                                                v++ ;
                                        }
                                 }

                        }
                    }
                    fclose(fp1);
                }
                shiftflag=0;
}
```

EDITOR SCREEN



CODE OPENED IN THE EDITOR