ST. ANNE'S SR. SEC SCHOOL, JODHPUR

Computer Science Project

Personal diary

By :- Harsh Uttamchandani

Class :- XIIth-A

Session :- 2019-20

Submitted To :- Mrs. Archana Agarwal

Board Roll no. :-

CERTIFICATE

This is to certify that Harsh Uttamchandani of class twelve, St. Anne's Senior Secondary School, Jodhpur has successfully completed his project in computer practicals on topic 'Personal diary' as prescribed by CBSE in the year 2019-2020.

Signature of Internal Examiner
MRS.ARCHANA AGRAWAL
[PGT COMPUTER SCIENCE]

Signature of External Examiner

Index

Acknowledgement

Firstly I would like to thank our CBSE for providing the opportunity to undertake this project. Next I would like to thank our computer teacher Mrs. Archana Agarwal for her immense guidance and support. I would also like to thank our principal Mr. John Abraham for providing us all the required infrastructure and facilities.

Lastly I would like to thank my parents and brother for collaborating me during my project hours at home. Their keen interest has really helped in making the project more interactive and user-friendly.

Introduction

Personal Diary Management System is based on the concept to save and generate all the diaries of the user. At first, the user has to pass through login system then the user can Add, view ,remove the diaries and also add reminders. Reminders are shown by checking dates if you login on the day of reminder.

A personal diary which most of the community uses and as everybody is moving from manual to the computers. It is another initiative to bring your own personal diary to your device.And many people are not good with dates so the adding reminder helps them.
These were the basic thought that made me to undertake this as my project.

There is also class 11 project car racing which is just fun whenever you enter the software and want to freshen up a little bit.

The whole project is designed in 'C++' language. This project is easy to operate and understand by the users.

Talking about technical aspects, the project fulfills the directives of CBSE. It implements modularity (by using functions), polymorphism (by using function overloading), abstraction (by using classes), Data file handling and all other aspects of a good program.

Abstract

Member Functions:

1) Class diary_entry

- Constructor and destructor
- Function to add diary entry

  void input();
- Function to display diary entries

  void display();
- Function to return date

  struct dosdate_t return_date();

2) Class reminders

- Constructor and destructor
- Function to add reminder

  void rem_input();
- Function to display reminder

  void rem_display();
- Function to return date

  struct dosdate_t rem_return_date();

3) Class User

- Constructor and destructor
- Function to add register

  int registinfo();

Non-Member Functions:

1)For Diary entries

- Function to write over file.
  void putdata()
- Functions to display all data.
  void getdata()
- Functions for removing information.
  void remove(struct dosdate_t dat)
- Function for viewing particular info
  void view_info(struct dosdate_t dat)

2)For Reminders

- Function to write over file.
  void rem_putdata()
- Functions to display all data.
  void rem_getdata()
- Functions for checking reminders.
  void check_reminders()

3)For User

- Function to write over file.
  void rem_putdata()
- Functions to display all data.
  void rem_getdata()
- Functions for checking reminders.
  void check_reminders()

4)For Menu display

- Function to display diary management menu.
  void menudisplay()
- Functions to display login page.
  void menudisplay2()
- Functions for checking reminders.
  void check_reminders()

5)For Car racing game

- Function to go display game's menu
  void start()
- Functions to create the graphics for game
  void soundcar()
  void collide_sound()
  void display_level(int level)
  int display_score(int score)
  void hide_displaylevel( )
  void green_court()
  void racing_track()
  void divider1()
  void drawcar1() etc
- Function for returning to main menu if game over.
  void GAMEOVER()

Requirements

Hardware and Software Requirements

HARDWARE REQUIRED

- Compact Drive
- Processor : Pentium III or higher
- RAM: 256 MB
- Harddisk : 20 Gb.

SOFTWARE REQUIRED

- Operating system : Windows XP or higher
- Turbo C++, for execution of program
- Adobe Reader, for User Guide and other documents

User Guide

Personal Diary
User Guide

Setting up the project:

1)Copy folder into your computer from the CD.

2)Open the file project MAIN.CPP in TURBO C++ IDE.

3)In the file on top you will find #define statements involving paths of graphics, header files etc. Give the paths correctly; Just change initial path to the path to which you have copied the folder to.

4)Press CTRL+F9 to run the program.

Input and Output Design

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
1. Personal Diary
2. Funtime:Car Racing
3. Exit
Enter your Choice - 1_
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
                    Personal Diary


1. Register as a New User
2. Login
3. Exit
Enter your Choice - 1
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
Enter Username - harsh

Enter Password - *****

Enter the Password again - *****
REGISTRATION SUCESSFULL_
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
                        Personal Diary


1. Register as a New User
2. Login
3. Exit
Enter your Choice - 2_
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
Enter Username - harsh

Enter Password - *****Credentials matched!
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
File could not be open.No Reminders Added
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
                    Personal Diary

01.ENTER DIARY

02.DELETE THE DIARY

03.DISPLAY ALL DIARY

04.DISPLAY THE REQUIRED DIARY

05.ENTER REMINDER

06.Exit

Enter the choice:1
new space created...Start writing your diary(press esc to exit)-:
hello my first diary.←
saving your working space
press any key to continue

            DO YOU WANT TO CONTINUE......????

            IF YES PRESS Y OR y:y
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
                    Personal Diary

01.ENTER DIARY

02.DELETE THE DIARY

03.DISPLAY ALL DIARY

04.DISPLAY THE REQUIRED DIARY

05.ENTER REMINDER

06.Exit

Enter the choice:3
19/11/2019
hello my first diary.

            DO YOU WANT TO CONTINUE......????

            IF YES PRESS Y OR y:_
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
                    Personal Diary

01.ENTER DIARY

02.DELETE THE DIARY

03.DISPLAY ALL DIARY

04.DISPLAY THE REQUIRED DIARY

05.ENTER REMINDER

06.Exit

Enter the choice:4
enter day 19
enter month 11
enter year 2019
19/11/2019
hello my first diary.
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
01.ENTER DIARY

02.DELETE THE DIARY

03.DISPLAY ALL DIARY

04.DISPLAY THE REQUIRED DIARY

05.ENTER REMINDER

06.Exit

Enter the choice:5
enter day 19
enter month 11
enter year 2019
new space created...Start writing your reminder(press esc to exit)-:
TEST TOMM.←
saving your working space
press any key to continue

                DO YOU WANT TO CONTINUE......????

                IF YES PRESS Y OR y:Y
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
                    Personal Diary

01.ENTER DIARY

02.DELETE THE DIARY

03.DISPLAY ALL DIARY

04.DISPLAY THE REQUIRED DIARY

05.ENTER REMINDER

06.Exit

Enter the choice:6_
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
                    Personal Diary


1. Register as a New User
2. Login
3. Exit
Enter your Choice - 2_
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

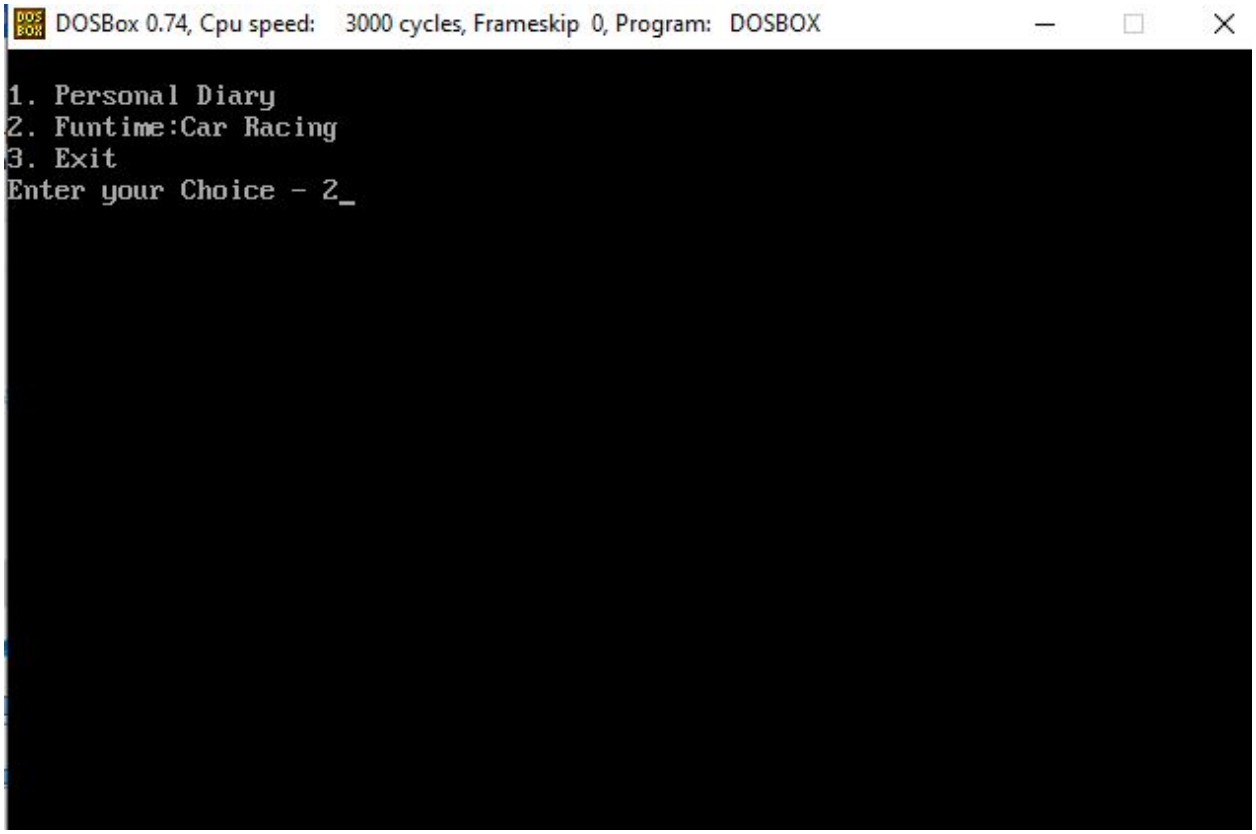
```
Enter Username - harsh

Enter Password - *****Credentials matched!
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
                    REMINDERS

19/11/2019
TEST TOMM._
```

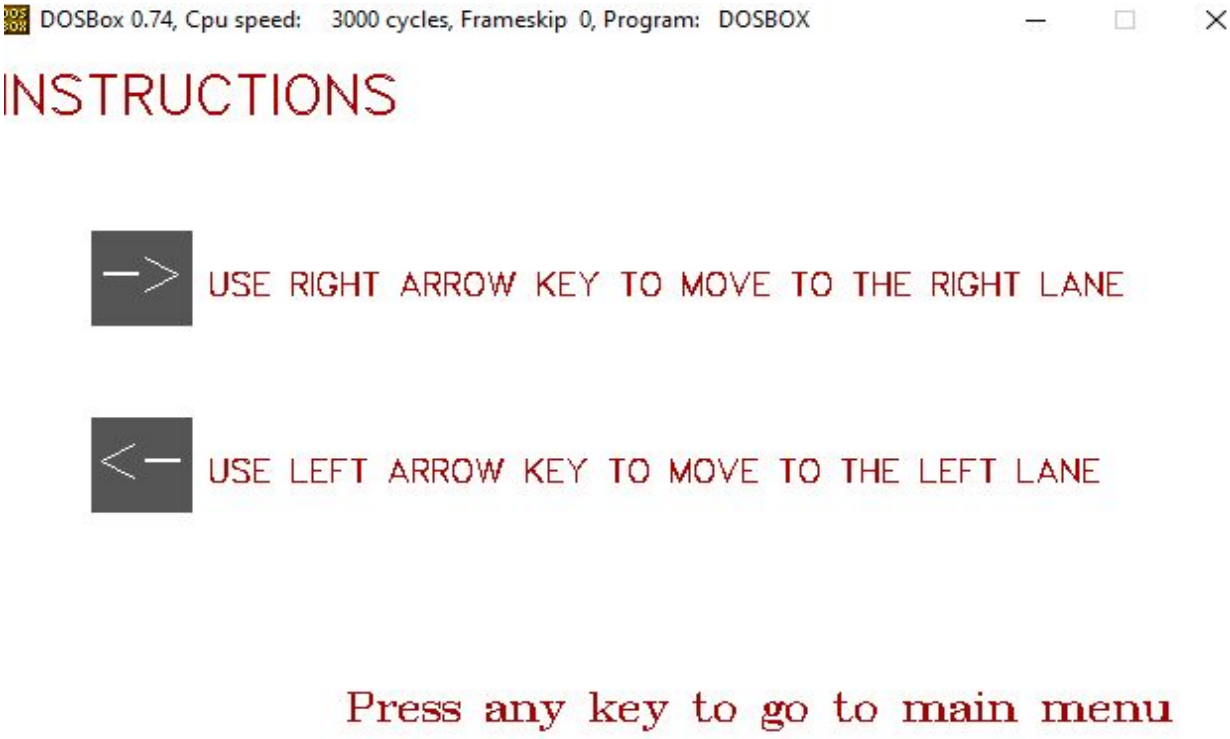

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
1. Personal Diary
2. Funtime:Car Racing
3. Exit
Enter your Choice - 2_

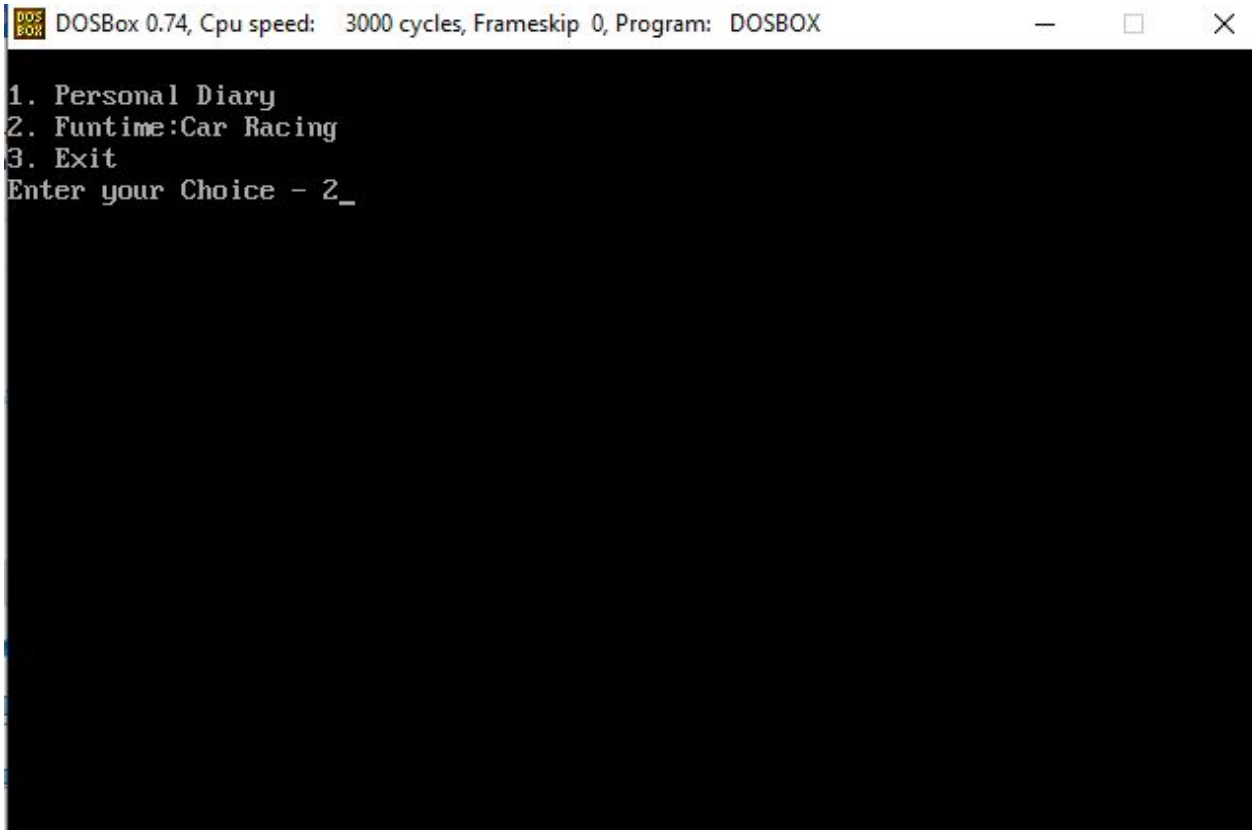

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
ROAD RUNNER
START
HELP
EXIT


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
ROAD RUNNER
START
HELP
EXIT

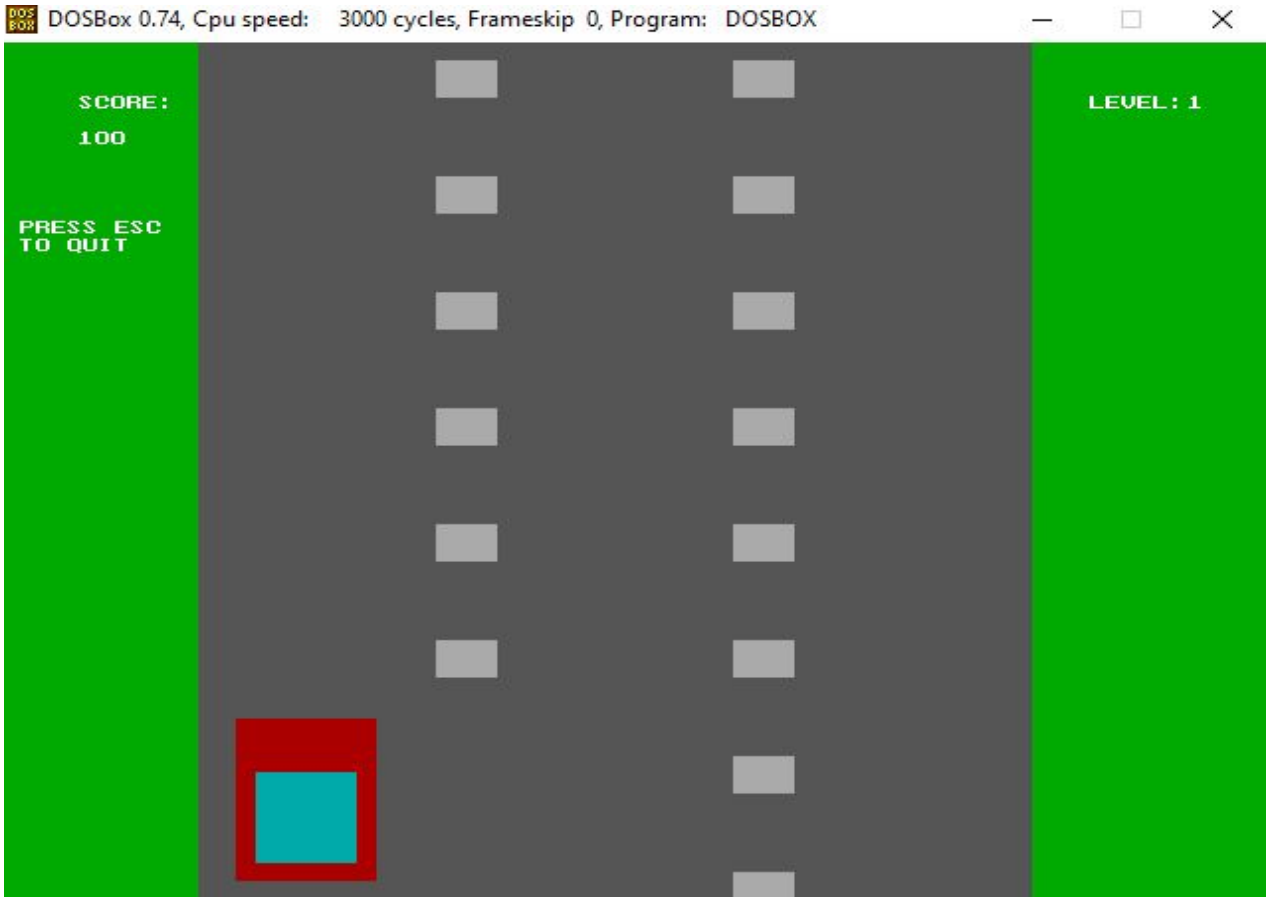

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
ROAD RUNNER
START
HELP
EXIT


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
ROAD RUNNER
START
HELP
EXIT


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
INSTRUCTIONS
USE RIGHT ARROW KEY TO MOVE TO THE RIGHT LANE
USE LEFT ARROW KEY TO MOVE TO THE LEFT LANE
Press any key to go to main menu


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
1. Personal Diary
2. Funtime:Car Racing
3. Exit
Enter your Choice - 2_


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
ROAD RUNNER
START
HELP
EXIT


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
SCORE:
100
PRESS ESC
TO QUIT
LEVEL: 1


DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
SCORE:
300
PRESS ESC
TO QUIT
GAME OVER
LEVEL: 3

Source Code

```
/*****************************************************
            HEADER FILES INCLUDED IN PROJECT
*****************************************************/
#include<fstream.h>
#include<conio.h>
#include<graphics.h>
#include<string.h>
#include<stdio.h>
#include<process.h>
#include<stdlib.h>
#include <dos.h>
#include<time.h>
/*****************************************************
            SOME FUNCTIONS DEFINITION
*****************************************************/

void start();
void mainmenu();
/*****************************************************
                        MENUS
*****************************************************/

int menu()
{
        int ch;
        system("cls");
        cout << "\n1. Personal Diary";
        cout << "\n2. Funtime:Car Racing";
```

```
	cout << "\n3. Exit";
	cout << "\nEnter your Choice - ";
	cin >> ch;
	return ch;
}

int menu1()
{
	int ch;
	system("cls");
	cout << "\t\t\tPersonal Diary \n\n";
	cout << "\n1. Register as a New User";
	cout << "\n2. Login";
	cout << "\n3. Exit";
	cout << "\nEnter your Choice - ";
	cin >> ch;
	return ch;
}

int menu2()
{
	int ch;
	system("cls");
		cout << "\t\t\tPersonal Diary \n\n";
		cout << "01.ENTER DIARY\n\n";
		cout << "02.DELETE THE DIARY\n\n";
		cout << "03.DISPLAY ALL DIARY \n\n";
		cout << "04.DISPLAY THE REQUIRED
DIARY\n\n";
```

```
            cout << "05.ENTER REMINDER\n\n";
            cout << "06.Exit\n\n";
            cout << "Enter the choice:";
        cin >> ch;
        return ch;
}

/*****************************************************************
CAR RACING:DECLARATIONS FOR STYLES AND
GRAPHICS
*****************************************************************/

#define maxx getmaxx()
#define maxy getmaxy()
#define midx maxx/2
#define midy maxy/2
#define bkcolor setbkcolor(4)
#define fillcol1 setcolor(col1)
#define fillcol2 setcolor(col2)
#define fillcol3 setcolor(col3)
#define left   midx+20
#define top    midx- 115
#define right  midx - 45
#define bottom  midx - 95

#define court_style setfillstyle(1,2)
#define racetrack_style setfillstyle(1,8)
#define divider_show setfillstyle(1,7)
#define loop1  r=0;r<455;r+=30
```

```
/*****************************************************
CAR RACING:ABOUT/INSTRUCTION FUNCTIONS1
*****************************************************/
void about()
{
        cleardevice();
        setbkcolor(15);
        settextstyle(3,HORIZ_DIR,4);
        outtextxy(5,5,"INSTRUCTIONS");

    setfillstyle(1,8);
        bar(100,100,50,150);
        settextstyle(3,HORIZ_DIR,4);
        setcolor(15);

        outtextxy(55,100,"->");
                settextstyle(3,HORIZ_DIR,1);
        setcolor(RED);
        outtextxy(110,115,"USE RIGHT ARROW KEY TO
MOVE TO THE RIGHT LANE");

        setfillstyle(1,8);
        bar(100,200,50,250);
        settextstyle(3,HORIZ_DIR,4);
        setcolor(15);

        outtextxy(55,200,"<-");
                settextstyle(3,HORIZ_DIR,1);
```

```
        setcolor(RED);
        outtextxy(110,215,"USE LEFT ARROW KEY TO MOVE
TO THE LEFT LANE");



        settextstyle(1,HORIZ_DIR,3);
        outtextxy(midx-140,midy+100,"Press any key to go to
main menu");


}
/*****************************************************************
CAR RACING:FUNCTION GAMEOVER
*****************************************************************/

void GAMEOVER()
{
        racetrack_style; //racing track
        bar(100,0,520,maxy);

        settextstyle(3,HORIZ_DIR,4);
        setcolor(WHITE);
        outtextxy(midx-80,midy-30,"GAME OVER");
        nosound();
    sound(450);
        delay(500);
        sound(260);
        delay(500);
```

```
        sound(550);
    delay(500);
        sound(200);
        delay(500);
        sound(200);
        delay(500);
    nosound();
        delay (700);
        start();

}
/*****************************************************************
CAR RACING:FUNCTIONS FOR SOUND,COLLIDE CHECK
AND LEVELS
*****************************************************************/

void graphics_initialize()
{
/* request auto detection */
int gdriver = DETECT, gmode;

/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "g:\\bgi");
}



void soundcar()
{
```

```
    sound(60);
    sound(80);
    sound(100);


}

void collide_sound()
{
        sound(1500);
    delay(210);
    nosound();
    sound(350);
    delay(150);
    nosound();
    sound(1000);
    delay(300);
        nosound();
        nosound();
}

void display_level(int level)
{
        char string2[25];
        itoa(level,string2,10);
        setcolor(WHITE);
        outtextxy(600,30,string2);

}
```

```
void hide_displaylevel( )
{
	setfillstyle(1,GREEN);
	bar(600,30,620,70);
}

int display_score(int score)
{
	char string[25];
	itoa(score,string,10);
	setcolor(WHITE);
	outtextxy(40,50,string);

return score;
}

void hide_display_score()
{
	setfillstyle(1,GREEN);
	bar(40,50,70,70);
}


void racing_track()
{
 racetrack_style; //racing track
 bar(100,0,520,maxy);
}
```

```
void green_court()
{ court_style; //green court
  bar(0,0, maxx,maxy);
}

void divider1()
{
bar(220,30,250,50);
bar(220,30+65,250,50+65);
bar(220,30+130,250,50+130);
bar(220,30+195,250,50+195);
bar(220,30+260,250,50+260);
bar(220,30+325,250,50+325);
bar(220,30+390,250,50+390);
bar(220,30+455,250,50+455);
}

void divider2()
{
bar(220,10,250,30);
bar(220,10+65,250,30+65);
bar(220,10+130,250,30+130);
bar(220,10+195,250,30+195);
bar(220,10+260,250,30+260);
bar(220,10+325,250,30+325);
bar(220,10+390,250,30+390);
bar(220,10+455,250,30+455);
}
```

```
void divider3()
{
bar(370,30,400,50);
bar(370,30+65,400,50+65);
bar(370,30+130,400,50+130);
bar(370,30+195,400,50+195);
bar(370,30+260,400,50+260);
bar(370,30+325,400,50+325);
bar(370,30+390,400,50+390);
bar(370,30+455,400,50+455);
}

void divider4()
{
bar(370,10,400,30);
bar(370,10+65,400,30+65);
bar(370,10+130,400,30+130);
bar(370,10+195,400,30+195);
bar(370,10+260,400,30+260);
bar(370,10+325,400,30+325);
bar(370,10+390,400,30+390);
bar(370,10+455,400,30+455);
}

void drawdivider1()
{
        divider_show;
        divider1();
        divider3();
```

```
}
void hidedivider1()
{
        racetrack_style;  //to hide the divider
        divider1();
        divider3();
}
void drawdivider2()
{
        divider_show;
        divider2();
        divider4();
}
void hidedivider2()
{
        racetrack_style;  //to hide the divider
        divider2();
        divider4();
}
void enemycar1(int r)
{
    setfillstyle(1,9);
    bar(140,-25+r,170,-5+r);
    setfillstyle(1,11);
    bar(150,-15+r,160,-10+r);

}
void enemycar2(int r)
{   setfillstyle(1,6);
```

```
        bar(300,-25+r,330,-5+r);
        setfillstyle(1,12);
        bar(310,-15+r,320,-10+r);


}

void enemycar3(int r)
{       setfillstyle(1,5);
        bar(450,-25+r,480,-5+r);
        setfillstyle(1,2);
        bar(460,-15+r,470,-10+r);

}
void erase_enemycar1(int r)
{
        setfillstyle(1,8);
        bar(140,-25+r,170,-5+r);

}
void erase_enemycar2(int r)
{       setfillstyle(1,8);
        bar(300,-25+r,330,-5+r);

        }

void erase_enemycar3(int r)
{
        setfillstyle(1,8);
```

```
        bar(450,-25+r,480,-5+r);
}

int collision(int x,int r,int lane,int lane2)
{
        int car[3],enemycar[2];
        car[1]=maxy-100;
        enemycar[1]=-25+r;
        enemycar[2]=-5+r;
        if((lane==x||lane2==x )&& (car[1] < enemycar[1] ||
car[1] < enemycar[2]) )
        {
           collide_sound();
       delay(200);
           GAMEOVER();
       return 1;
        }
return 0;
}

void drawcar1()
{
   setfillstyle(1,4);
   bar(midx-200, maxy-100,midx-130,maxy-10);
   setfillstyle(1,3);
   bar(midx-190, maxy-70, midx-140,maxy-20);
   setfillstyle(1,7);
}
void drawcar2()
```

```
{
  setfillstyle(1,4);
  bar(midx-40, maxy-100,midx+30,maxy-10);
  setfillstyle(1,3);
  bar(midx-30, maxy-70, midx+20,maxy-20);
  setfillstyle(1,7);
}
void drawcar3()
{
  setfillstyle(1,4);
  bar(midx+110, maxy-100,midx+180,maxy-10);
  setfillstyle(1,3);
  bar(midx+120, maxy-70, midx+170,maxy-20);
  setfillstyle(1,7);
}


void erasecar1()
{
      setfillstyle(1,8);
   bar(midx-200, maxy-100,midx-130,maxy-10);
   setfillstyle(1,3);

}

void erasecar2()
{
  setfillstyle(1,8);
  bar(midx-40, maxy-100,midx+30,maxy-10);
```

```
}
void erasecar3()
{
  setfillstyle(1,8);
  bar(midx+110, maxy-100,midx+180,maxy-10);

}

int check(int x)
{
        if (kbhit())
        {
        int a=getch();
                if(a==27)
                {
                        nosound();
                exit(0);
                }

                if(a=='M')
                        {if (x==2)
                                {
                                sound(1200);
                delay(20);
                                nosound();
                                soundcar();

                                drawcar3();
```

```
            erasecar2();
            x=3;
            }
        if(x==1)
            {
             sound(1200);
 delay(20);
             nosound();
             soundcar();
             drawcar2();
             erasecar1();
             x=2;
            }
        }
if(a=='K')
{
        if(x==2)
        {sound(1200);
 delay(20);
             nosound();
             soundcar();
             drawcar1();
             erasecar2();
             x=1;
        }
        if(x==3)
        {
             sound(1200);
 delay(20);
```

```
                    nosound();
                    soundcar();
                drawcar2();
                erasecar3();
                x=2;

                }
            }

        }
return x;
}

int random_carlane(int lane)
{
  randomize();
  lane=random(100)+1;
      if(lane%2==0)
            lane = 2;
      else
      {if(lane%3)
            lane =3;
      else
            lane=1;
   }
  return lane;
}

int random_carlane2(int lane2,int lane)
```

```
{
        randomize();
        lane2=random(70)+1;
        if(lane2%2==0)
                lane2=1;
        else
        {if(lane2==lane)
                        lane2=1;
                else
                        lane2=3;
        }
   return lane2;
}
void drawrec1(int col1)
{
fillcol1;
rectangle(left,top,right,bottom);
}
void drawrec2(int col2)
{
fillcol2;
rectangle(left,top+30,right,bottom+30);
}
void drawrec3(int col3)
{
fillcol3;
rectangle(left,top+60,right,bottom+60);
}
```

```
/*****************************************************
CAR RACING:FUNCTION FOR DRAWING THE LANES
*****************************************************/
void intro()
{
int r=0,ch=0,lane,lane2,level=1;
int x=2,score=0,tt=260;
int collide=0;
green_court();//draws green court
racing_track();//draws racing track
setcolor(WHITE);
outtextxy(40,30,"SCORE:");
outtextxy(10,100,"PRESS ESC");
outtextxy(10,110,"TO QUIT");
outtextxy(550,30,"LEVEL:");
do
{

hide_displaylevel();
display_level(level);

hide_display_score();
        display_score(score);
if(ch==0)
   {
            drawcar2();
   }
    x=check(x);
       lane=random_carlane(lane);
```

```
        lane2=random_carlane2(lane2,lane);
soundcar();
        for(loop1)
{
     drawdivider1();
        delay(tt);

        x= check(x);
        hidedivider1();
        drawdivider2();
        x=check(x);
        if(lane==1||lane2==1)
        enemycar1(r);
        if(lane==2||lane2==2)
        enemycar2(r);
        if(lane==3||lane2==3)
        enemycar3(r);
        delay(tt);
        score+=10;
        hide_display_score();
        display_score(score);
        x=check(x);


        if(r<500)
         {
                        if(lane==1||lane2==1)
                        erase_enemycar1(r);
                        if(lane==2||lane2==2)
```

```
                    erase_enemycar2(r);
                    if(lane==3||lane2==3)
                    erase_enemycar3(r);
                    collide=collision(x,r,lane,lane2);
                    x=check(x);

            }
if(score%140==0 && tt>20)
{tt-=15;
  x=check(x);
level+=1;
hide_displaylevel();
display_level(level);
}
x=check(x);
        hidedivider2();
         x=check(x);



}
ch=1;

}
while(!collide);

}
```

```
/*****************************************************
CAR RACING:FUNCTION FOR MAIN MENU DISPLAY
*****************************************************/


void start()
{
        graphics_initialize();
int midxx,midyy,c,i=0,col1,col2,col3,j=0,k=0;

midxx = midx-30;
midyy = midy-30;
bkcolor;
setcolor(15);
outtextxy(midxx,midyy,"START");
outtextxy(midxx,midyy+30,"HELP");
outtextxy(midxx,midyy+60,"EXIT");
settextstyle(4,HORIZ_DIR,7);
setcolor(14);
outtextxy(midxx-225,midyy-200,"ROAD RUNNER");
settextstyle(0,HORIZ_DIR,1);
do
{       c=getche();
                if(c==80 && i==0 || c==72 && j==1)
                        {       col1=2;
                                drawrec1(col1);
                                i=1;k=1;
                                col2=4;
                                col3=4;
```

```
                    drawrec2(col2);
                    drawrec3(col3);
                    continue;
                }
            if(c==80 && i==1 || c==72 && j==2)
                {
                 col2=2;
                 drawrec2(col2);
                 i=2;j=1;k=2;
                 col1=4;
                 col3=4;
                drawrec1(col1);
                drawrec3(col3);
                continue;
                }
if(c==80 && i==2 || c==72 && j==0)
{
col3=2;
drawrec3(col3);
i=3;j=2;k=3;
col2=4;
col3=4;
drawrec1(col1);
drawrec2(col2);
continue;
}
}
while(c!=13);
switch(k)
```

```
 { case 1 :intro();break;
   case 2 :about();break;
   case 3 :mainmenu();break;
}


getch();
closegraph();
}


/*****************************************************************
PERSONAL DIARY SYSTEM:GLOBAL VARIABLE IN
PROJECT
*****************************************************************/
int wordlimit;
/*****************************************************************
PERSONAL DIARY SYSTEM:CLASSES USED IN PROJECT
*****************************************************************/

/*****************************************************************
PERSONAL DIARY SYSTEM:CLASS DIARY_ENTRY
*****************************************************************/
class diary_entry
{

        struct dosdate_t d;
        char diary[1000];
        public :
```

```
        diary_entry();
        void display();
        void input();
        struct dosdate_t return_date();
        ~diary_entry();

};



//constructor
            diary_entry::diary_entry()
                {
                    int i=0;
                    wordlimit=0;
                    for (i=0;i<1000;i++)
                        diary[i]='\0';
                }

//For input
void diary_entry :: input()
{

   char ch;
   _dos_getdate(&d);
   cout<<"new space created...Start writing your
diary(press esc to exit)-:"<<'\n';
   do
   {
```

```
ch=getche();
if(ch!=13 && ch!=8 && ch!=27)
{
   diary[wordlimit]=ch;
   wordlimit++;
}
else
{
   if(ch==13)
   {
      cout<<'\n';
      diary[wordlimit]='\n';
      wordlimit=wordlimit+1;
   }
   else if(ch==8)
   {
      diary[wordlimit]='\b';
      wordlimit--;
      cout<<' ';
      cout<<'\b';
   }
   if(wordlimit>=999)
   {
      diary[wordlimit]='\0';
      cout<<"limit exceded";
      getch();
   }
}
if(ch==27)
```

```
        {
            diary[wordlimit]='\0';

            cout<<'\b';
            cout<<'\n'<<"saving your working space";
            cout<<'\n'<<"press any key to continue";

            getch();
        }
    }
    while(ch!=27 && wordlimit<1000);

}



//To compare date
        struct dosdate_t diary_entry :: return_date()
        {

            return d ;
        }

//Display simply
        void diary_entry :: display()
        {
                char day[3] , month[3];
                itoa(d.day, day, 10);
                itoa(d.month, month, 10);
                cout<<day <<'/'<< month <<'/'<<d.year<<'\n';
```

```
            cout<<diary;

        }

        diary_entry :: ~diary_entry()
            {
                ;
            }


/*****************************************************************
                PERSONAL DIARY SYSTEM:FUNCTIONS
*****************************************************************/

//to write over file
void putdata()
{
   diary_entry c;
   ofstream fout;
   fout.open("diary.dat",ios::binary|ios::app);
   c.input();
   fout.write((char*)(&c),sizeof(c));
   fout.close();
}



//to display all data
void getdata()
```

```
{
    diary_entry c;
    ifstream fin;
    fin.open("diary.dat",ios::binary,ios::beg);
    while(fin.read((char*)(&c),sizeof(c)))
    {
        c.display();
    }
    fin.close();
}


//remove information
void remove(struct dosdate_t dat)
{

    diary_entry obj;
    int flag;
    flag=0;
    ifstream fin;
    ofstream fout;
    fin.open("diary.dat",ios::binary);
    if(!fin)
    {
        cout<<"File could not be open ";
        return;
    }
    fout.open("copy.dat",ios::binary);
    fin.seekg(0,ios::beg);
```

```
    struct  dosdate_t dat1;

    while(fin.read((char *)(&obj), sizeof(diary_entry)))
    {
        dat1=obj.return_date();
        if(dat1.day==dat.day && dat1.month==dat.month
&& dat1.year==dat.year)
        {
            flag=1;
        }
        if(dat1.day!=dat.day || dat1.month!=dat.month ||
dat1.year!=dat.year)
        {
            fout.write((char *)(&obj), sizeof(diary_entry));
        }
    }
    fout.close();
    fout.close();
    remove("diary.dat");
    rename("copy.dat","diary.dat");
    if(flag==0)
    {
        cout<<"\n\n\tNo records found ..";
    }


    if(flag==1)
    {
        cout<<"\n\n\tRecords Removed ..";
```

```
    }
    getch();

}

//view particular info
void view_info(struct dosdate_t dat)
{

    int flag;
    diary_entry obj;
    flag=0;
    ifstream fin;
    fin.open("diary.dat",ios::binary);
    if(!fin)
    {
        cout<<"File could not be open.";
        return;
    }
    struct  dosdate_t dat1;
    fin.seekg(0,ios::beg);
    while(fin.read((char *)(&obj), sizeof(diary_entry)))
    {
        dat1=obj.return_date();
        if(dat1.day==dat.day && dat1.month==dat.month
&& dat1.year==dat.year)
        {

            obj.display();
```

```
            flag=1;
        }
    }
    fin.close();
    if(flag==0)
        cout<<"\n\n                    Information does not exist";
    getch();
}

struct dosdate_t get_date()
{
        int dd;
        int mm;
        int yy;
        cout<<"enter day ";
        cin>>dd;
        cout<<"enter month ";
        cin>>mm;
        cout<<"enter year ";
        cin>>yy;
        struct dosdate_t d;
        d.day=dd;
        d.month=mm;
        d.year=yy;
        return d;

}
```

```
/*****************************************************************
PERSONAL DIARY SYSTEM: CLASS REMINDERS
*****************************************************************/
class reminders
{

   struct dosdate_t r;
   char reminder[1000];
public :

   reminders();
   void rem_display();
   void rem_input();
   struct dosdate_t rem_return_date();
   ~reminders();

};


//constructor
reminders::reminders()
{
   int i=0;
   wordlimit=0;
   for(i=0; i<1000; i++)
   {
      reminder[i]='\0';
   }
}
```

```
//FOR INPUT
void reminders :: rem_input()
{
    char ch;
    int dd;
    int mm;
    int yy;
    cout<<"enter day ";
    cin>>dd;
    cout<<"enter month ";
    cin>>mm;
    cout<<"enter year ";
    cin>>yy;
    r.day=dd;
    r.month=mm;
    r.year=yy;

    cout<<"new space created...Start writing your
reminder(press esc to exit)-:"<<'\n';
    do
    {
        ch=getche();
        if(ch!=13 && ch!=8 && ch!=27)
        {
            reminder[wordlimit]=ch;
            wordlimit++;
        }
        else
```

```
{
   if(ch==13)
   {
      cout<<'\n';
      reminder[wordlimit]='\n';
      wordlimit=wordlimit+1;
   }
   else if(ch==8)
   {
      reminder[wordlimit]='\b';
      wordlimit--;
      cout<<' ';
      cout<<'\b';
   }
   if(wordlimit>=999)
   {
      reminder[wordlimit]='\0';
      cout<<"limit exceded";
      getch();
   }
}
if(ch==27)
{
   reminder[wordlimit]='\0';

   cout<<'\b';
   cout<<'\n'<<"saving your working space";
   cout<<'\n'<<"press any key to continue";
```

```
            getch();
        }
    }
    while(ch!=27 && wordlimit<1000);

}

//To return date
struct dosdate_t reminders :: rem_return_date()
{

    return r;
}

//Display simply
void reminders :: rem_display()
{
    char day[3], month[3];
    itoa(r.day, day, 10);
    itoa(r.month, month, 10);
    cout<<day <<'/'<< month <<'/'<<r.year<<'\n';
    cout<<reminder;
}

reminders :: ~reminders()
{
    ;
}
```

```
/*****************************************************************
PERSONAL DIARY SYSTEM: FUNCTIONS REMINDERS
*****************************************************************/

//to write over file
void rem_putdata()
{
   reminders c;
   ofstream fout;
   fout.open("reminder.dat",ios::binary|ios::app);
   c.rem_input();
   fout.write((char*)(&c),sizeof(c));
   fout.close();
}



//to display all data
void rem_getdata()
{
   reminders c;
   ifstream fin;
   fin.open("reminder.dat",ios::binary,ios::beg);
   while(fin.read((char*)(&c),sizeof(c)))
   {
      c.rem_display();
   }
```

```
    fin.close();
}


void check_reminders()
{
    clrscr();
    struct dosdate_t dat;
    _dos_getdate(&dat);


    int flag;
    reminders obj;
    flag=0;
    ifstream fin;
    fin.open("reminder.dat",ios::binary);
    if(!fin)
    {
        cout<<"File could not be open.No Reminders Added";
        return;
    }
    struct  dosdate_t dat1;
    fin.seekg(0,ios::beg);
    cout<<"\t\t\t REMINDERS\n\n";
    while(fin.read((char *)(&obj), sizeof(reminders)))
    {
        dat1=obj.rem_return_date();
        if(dat1.day==dat.day && dat1.month==dat.month
&& dat1.year==dat.year)
```

```
        {

            obj.rem_display();
            flag=1;
        }
    }
    fin.close();
    if(flag==0)
        cout<<"\n\n                    No Reminders Today";
    getch();

}
/*****************************************************************
PERSONAL DIARY SYSTEM :Menu display 1
*****************************************************************/
void menudisplay()
{
    int ch;
    char cho;
    check_reminders();
    getch();
    do
    {
        clrscr();
        ch=menu2();
        switch(ch)
        {
        case 1 :
            putdata();
```

```
        break;
    case 2 :
        struct dosdate_t dat=get_date();
        remove(dat);
        break;
    case 3 :
        getdata();
        break;
    case 4 :
        struct dosdate_t temp=get_date();
        view_info(temp);
        break;
    case 5 :
        rem_putdata();
        break;

    case 6 :
        exit(0);
        break;
    default :
        continue;
    }
    cout << "\n\n\t\tDO YOU WANT TO
CONTINUE.....!!!!";
    cout << "\n\n\t\tIF YES PRESS Y OR y:";
    cin >> cho;
  }
  while(cho == 'y' || cho == 'Y');
}
```

```
/*****************************************************
PERSONAL DIARY SYSTEM:CLASS USER
*****************************************************/
class user
{
public:
   char usrnm[40];
   char pswrd[8];
   char psword[8];

   user()
   {
      strcpy(usrnm, "NULL");
      strcpy(pswrd,"NULL");
      strcpy(psword,"NULL");
   }

   ~user()
   {
      ;
   }

   int registinfo();
};
int user::registinfo()
{
   cout << "\nEnter Username - ";
```

```
cin >> usrnm;
cout << "\nEnter Password - ";
char ch;
int i=0;
do
{

    ch=getch();
    if(ch!=13 && ch!=8)
    {
        pswrd[i]=ch;
        i++;
        cout<<'*';
    }
    else
    {
        if(ch==8)
        {
            if(i>0)
            {
                cout<<'\b';
                cout<<' ';
                cout<<'\b';
                i--;
            }
        }
        if(ch==13)
        {
            pswrd[i]='\0';
```

```
        }


    }
}
while(ch!=13);

cout << "\n\nEnter the Password again - ";

i=0;
do
{

    ch=getch();
    if(ch!=13 && ch!=8)
    {
        psword[i]=ch;
        i++;
        cout<<'*';
    }
    else
    {
        if(ch==8)
        {
            if(i>0)
            {
                cout<<'\b';
                cout<<' ';
                cout<<'\b';
```

```
                i--;
            }
        }
        if(ch==13)
        {
            psword[i]='\0';
        }

    }
  }
  while(ch!=13);

  if (strcmp(pswrd,psword)!=0)
  {
    cout << "\nPasswords Do not Match";
    getch();
    return -1;
  }

  return 1;
}

void registerinfo()
{
  system("cls");

  user u;

  ofstream fout;
```

```
    fout.open("userfile.dat", ios::binary);

    if (!fout)
        cout << "\nError in Opening File";
    else
    {

        if(u.registinfo()==1)
        {
            cout<<"\nREGISTRATION SUCESSFULL";
            fout.write((char *)&u, sizeof(u));
        }
    }
    fout.close();
}

void login()
{
    system("cls");

    char unm[40];
    char pd[8];
    user u;

    ifstream fin;
    fin.open("userfile.dat", ios::binary);

    if (!fin)
        cout << "\nError Opening File";
```

```
else
{
    cout << "\nEnter Username - ";
    cin >> unm;
    cout << "\nEnter Password - ";
    char ch;
    int i=0;
    do
    {

        ch=getch();
        if(ch!=13 && ch!=8)
        {
            pd[i]=ch;
            i++;
            cout<<'*';
        }
        else
        {
            if(ch==8)
            {
                if(i>0)
                {
                    cout<<'\b';
                    cout<<' ';
                    cout<<'\b';
                    i--;
                }
            }
```

```
            if(ch==13)
            {
                pd[i]='\0';
            }



        }
    }
    while(ch!=13);

    fin.seekg(0);

    while (fin.read((char*)&u, sizeof(u)))
    {
        if (strcmp(unm, u.usrnm) == 0)
        {
            if (strcmp(u.pswrd, pd) == 0)
            {
                cout << "Credentials matched!";
                getch();
                menudisplay();



            }
            else
            {
                cout << "\nPassword/Username Incorrect
Please try again";
                getch();
```

```
                login();
            }
        }
    }
    fin.close();
  }
}

/*****************************************************************
PERSONAL DIARY SYSTEM : Menu display 2
*****************************************************************/



void menudisplay2()
{
  clrscr();
  int ch;
  do
  {
    ch = menu1();

    switch (ch)
    {
    case 1:
      registerinfo();
      getch();
      break;
    case 2:
```

```
            login();
            getch();
            break;
        case 3:
            exit(0);
            break;
        default:
            cout << "Enter a valid choice";
            getch();
            break;
        }
    }
    while (ch != 3);
}
void mainmenu()
{
    clrscr();
    int ch;
    do
    {
        ch = menu();
        switch (ch)
        {
        case 1:
            menudisplay2();
            getch();
        case 2:
            start();
            getch();
```

```
        break;
      case 3:
        exit(0);
        break;
      default:
        cout << "Enter a valid choice";
        getch();
        break;
      }
    }
    while (ch != 3);
}
/*****************************************************************
                    MAIN OF PROGRAM
*****************************************************************/

void main()
{
    mainmenu();



}
```

Data Dictionary

Classes Used

1)Diary_entry

Contains following data members-:

-:struct dosdate_t d;

-:char diary[1000];

2)Reminders

Contains following data members-:

-:struct dosdate_t r;

-:char reminder[1000];

3)User

Contains following data members-:

-:char usrnm[40];

-:char pswrd[8];

-:char psword[8];

FILES USED

1) diary.dat

Binary file used to store date and diary entry ;object of class diary_entry.

2)reminder.dat

Binary file used to store date and diary reminder ;object of class reminders.

3)userfile.dat

Binary file used to store username and password ;object of class user.

Limitations and Enhancements

Limitations

- **The Program uses linear search algorithm which causes delay in searches if a lot of diary entries are there.**

- **There are some symbols/characters (though very few) in the definitions of the words which are not recognized in DOS environment. Hence unknown symbols are displayed.**

- **There is no date validation checks so user has to enter a valid format only.**

- **Due to lack of time no option for modifying diary and reminders could be added.**

Enhancements

- **File size can be made small by using a text file instead of binary.**

- **The Personal Diary System can be made for multi-user purpose by creating different diary and reminder file by using their user details.**

- **The function to modify the diary entries and reminders can be added.**

- **The Presentation of the project be more attractive by using more graphics functions.**

Bibliography

- Sumita Arora C++ [Class 11 and 12]