

Projects

A.1 MINOR PROJECT – COLLEGE MANAGEMENT SYSTEM (CMS)

Learning Objectives

After going through this project, you will be able to

- ❑ Develop menu-driven applications in C++
- ❑ Apply authentication in menu-driven projects
- ❑ Apply the file-handling concepts

College Management System (CMS)

CMS is a command-driven application that helps to maintain student and faculty records. It makes use of unique IDs and password to restrict the access and usage of different areas of the portal to its correct target audience. The faculty portal allows the faculty user to update faculty profile, including allocation of subjects to be taught. Likewise, the student portal helps to maintain student profile information, including allocation and de-allocation of subjects to the students. The admin portal is used for administrative purposes. It helps to make the user entry for the first time.

The CMS application code makes use of a wide array of C++ programming features. These include:

- Concept of inheritance for code reusability
- Concept of function overloading that enhances the code readability
- Handling of simple exceptions
- Application of file-handling concepts for performing operations, such as read, write, append, check for various I/O errors, etc.
- Use of various I/O manipulators to display the output in a formatted manner

Table A.1 lists the key coding elements of the CMS application:

Table A.1 Key Coding Elements

Element	Description
Admin	It is used to maintain the faculty and student profiles. It provides a unique login to both profile users, and allows them to update their records further.
getstddata()	It is used to make the student data entry from the ADMIN portal.

(Contd.)

Table A.1 (Contd.)

Element	Description
login()	It authenticates the login of both profile users (students and faculties).
recover()	It helps both profile users to recover the password.
addstsub()	It allows the student to add a new subject in his profile.
delstsub()	It allows the student to delete a subject from his profile.
modstprofile()	It allows the student to modify the personal profile details.
getfadata()	It allows the Admin to make a new entry of the faculty user.
stdisplay()	It displays all the student records to the admin.
fadisplay()	It displays all the faculty records to the admin.

Table A.2 depicts the overall logic used by the CMS application:

Table A.2 Application Logic

Stage	Logic
1	The Welcome Page of the CMS application is displayed at first. On hitting the Enter key, the user enters the main Login page. The user can login as Admin, Student or Faculty. <ul style="list-style-type: none"> • The Admin Panel is used to create and maintain the databases of all the faculty and student users. • The Faculty Panel enables access to all the registered faculties in the database. • The Student Panel enables access to all the registered students in the database.
2	Next, the user is distinguished as Admin, Student or Faculty based on the choice selected by the user. Subsequently, the user is redirected to the respective Login page.
3	Each of the respective portals are menu-driven. It shows a menu of operations that the users (faculty, student, admin) can perform. Refer the program output to view how the application functions and how the program flow is controlled.

Application Code

The code for CMS application is given below. It has comments at appropriate places to help understand the code better.

CMS APPLICATION

```
#include<iostream.h>
#include<stdlib.h>
#include<string.h>
#include<stdio.h>
#include<fstream.h>
#include<conio.h>
char un[20];
class Admin
{
    char name[20];
    int totsub;
```

(Contd.)

```

char subject[10][10];
char mobile[15],mail[50],fname[20];
char passwd[20],rpasswd[20];
public:
char rollno[15];
//this functin is used to get the student data entry from the ADMIN
portal....all the records entries are made by this method
void getstdata()
{
    cout<<"\nEnter the Student Name : ";
    gets(name);
    cout<<"\nEnter the Student ID/Roll No. : ";
    cin>>rollno;
    cout<<"\nEnter the Student Father's Name : ";
    gets(fname);
    cout<<"\nEnter the Mobile Number : ";
    gets(mobile);
    cout<<"\nEnter the E-Mail ID : ";
    gets(mail);
    cout<<"\nEnter the Total Subjects : ";
    cin>>totsub;
    for(int i=0;i<totsub;i++)
    {
        cout<<"\nEnter the Subject "<<i+1<<" Name : ";
        cin>>subject[i];
    }
    cout<<"\nCreate Your Login Password : ";
    cin>>passwd;
    cout<<"\nEnter the Unique Keyword to Recover Password : ";
    cin>>rpasswd;
    cout<<"\n\nPlease note your UserName is ID/Roll no.\n";
}
//this function authenticates the login of both the students & faculties.
With their reference to thier unique ID/PASSWORD login is provided
int login()
{
    if((strcmp(:un,rollno))==0)
    {
        cout<<"\nEnter the Login Password : ";
        int len=0;
        len=strlen(passwd);
        char inputpasswd[20];
        for(int i=0;i<len;i++)
        {
            inputpasswd[i]=getch();
            cout<<"#";
        }
        inputpasswd[i]=NULL;
        //cout<<"\nThe Entered Password is "<<inputpasswd;
        if((strcmp(:un,rollno)==0)&&(strcmp(passwd,inputpasswd)==0))
            return 1;
        else
            return 0;
    }
}

```

(Contd.)

```

        else
            return 0;
    }
    //this function help both the students & faculties to recover the password
    with the help of unique keyword provided by the Admin
    int recover()
    {
        //cout<<"\nThe Input Username is "<<::un;
        char key[20];
        if(strcmp(::un,rollno)==0)
        {
            cout<<"\nEnter the Unique Keyword ( Provided by Admin ) : ";
            cin>>key;
            if((strcmp(key, rpasswd)==0))
            {
                cout<<"\nYou are a Valid user." ;
                cout<<"\nYour Password is "<<passwd;
                cout<<"\nPlease Exit To Login Again ";
                return 1;
            }
            else
                return 0;
        }
        else
            return 0;
    }
    //this function displays the faculty profile to the faculty at their
    respective portals
    int faprofile()
    {
        if((strcmp(::un,rollno)==0))
        {
            cout<<"\nFaculty Name          : "<<name;
            cout<<"\nFaculty Father's Name : "<<fname;
            cout<<"\nFaculty Mobile No.    : "<<mobile;
            cout<<"\nFaculty E-Mail ID     : "<<mail;
            return 1;
        }
        else
            return 0;
    }
    //this function displays the different subjects of the faculty at thier
    respective portals
    int knowfasub()
    {
        if((strcmp(::un,rollno)==0))
        {
            cout<<"\nFaculty Total Subjects :- "<<totsub;
            for(int i=0;i<totsub;i++)
            {
                cout<<"\n\tSubject "<<i+1<<" : "<<subject[i];
            }
            return 1;
        }
    }

```

(Contd.)

```

        else
            return 0;
    }
    //this function helps a faculty to add a subject in their module.
    void addfasub()
    {
        if((strcmp(:un,rollno)==0))
        {
            cout<<"\nEnter the New Subject : ";
            cin>>subject[totsub];
            totsub++;
            cout<<"\n\nNew Subject Added Successfully...";
        }
    }
    //this fuction helps a faculty to delete a subject from their module.
    void delfasub()
    {
        knowfasub();
        int de=0;
        if((strcmp(:un,rollno)==0))
        {
            if(totsub==0||totsub<0)
            {
                totsub=0;
                cout<<"\nNone Subjects Exist...";
                getch();
                exit(0);
            }
            cout<<"\nEnter the Subject No. to be Deleted : ";
            cin>>de;
            if(de==totsub)
            {
                totsub--;
                strcpy(subject[totsub]," ");
            }
            else if(totsub==1)
            {
                totsub=0;
                strcpy(subject[totsub]," ");
            }
            else
            {
                de--;
                strcpy(subject[totsub]," ");
                for(int p=de;p<totsub;p++)
                {
                    strcpy(subject[p],subject[p+1]);
                }
                totsub--;
            }
            cout<<"\n Records Updated Successfully...";
        }
    }
    //this function helps a faculty to modify his personal profile

```

(Contd.)

```

void modfaprofile()
{
    if((strcmp(:un,rollno)==0))
    {
        cout<<"\nThe Profile Details are : ";
        cout<<"\n 1. Faculty E-Mail : "<<mail;
        cout<<"\n 2. Faculty Mobile : "<<mobile;
        int g=-1;
        cout<<"\n\nEnter the Detail No. to be Modified : ";
        cin>>g;
        if(g==1)
        {
            char nmail[50];
            cout<<"\nEnter the New Mail Address : ";
            strcpy(mail,nmail);
            cout<<"\nRecords Updated Successfully....";
        }
        else if(g==2)
        {
            char nmobile[15];
            cout<<"\nEnter the New Mobile No. : ";
            gets(nmobile);
            strcpy(mobile,nmobile);
            cout<<"\nRecords Updated Successfully...";
        }
        else
            cout<<"\nInvalid Input Provided...";
    }
}

//This function displays the student profile at the student portal
int stprofile()
{
    if((strcmp(:un,rollno)==0))
    {
        cout<<"\nStudent Name          : "<<name;
        cout<<"\nStudent Father's Name : "<<fname;
        cout<<"\nStudent Mobile No.      : "<<mobile;
        cout<<"\nStudent E-Mail ID       : "<<mail;
        return 1;
    }
    else
        return 0;
}

//this function displays the different enrolled subjects of the respective
students...according to their profile.
int knowstsub()
{
    if((strcmp(:un,rollno)==0))
    {
        cout<<"\nStudent Total Subjects :- "<<totsub;
        for(int i=0;i<totsub;i++)
        {
            cout<<"\n\tSubject "<<i+1<<" : "<<subject[i];
        }
    }
}

```

(Contd.)

```

        return 1;
    }
    else
        return 0;
}
//this function allows the student to add a subjects in total subjects of
the student profile...
void addstsub()
{
    if((strcmp(:un,rollno)==0))
    {
        cout<<"\nEnter the New Subject : ";
        cin>>subject[totsub];
        totsub++;
        cout<<"\n\nNew Subject Added Successfully...";
    }
}
//this function allows the student to delete a subjects in total subjects
of the student profile...
void delstsub()
{
    knowstsub();
    int de=0;
    if((strcmp(:un,rollno)==0))
    {
        if(totsub==0 || totsub<0)
        {
            totsub=0;
            cout<<"\nNone Subjects Exist...";
            getch();
            exit(0);
        }
        cout<<"\nEnter the Subject No. to be Deleted : ";
        cin>>de;
        if(de==totsub)
        {
            totsub--;
            strcpy(subject[totsub], " ");
        }
        else if(totsub==1)
        {
            totsub=0;
            strcpy(subject[totsub], " ");
        }
        else
        {
            de--;
            strcpy(subject[totsub], " ");
            for(int p=de;p<totsub;p++)
            {
                strcpy(subject[p], subject[p+1]);
            }
            totsub--;
        }
    }
}

```

(Contd.)

```

        cout<<"\n Records Updated Successfully...";
    }
}
//this function allows the student to modify their personal profile details
at the student portal
void modstprofile()
{
    if((strcmp(:un,rollno)==0))
    {
        cout<<"\nThe Profile Details are : ";
        cout<<"\n 1. Student E-Mail : "<<mail;
        cout<<"\n 2. Student Mobile : "<<mobile;
        int g=-1;
        cout<<"\n\nEnter the Detail No. to be Modified : ";
        cin>>g;
        if(g==1)
        {
            char nmail[50];
            cout<<"\nEnter the New Mail Address : ";
            strcpy(mail,nmail);
            cout<<"\nRecords Updated Successfully....";
        }
        else if(g==2)
        {
            char nmobile[15];
            cout<<"\nEnter the New Mobile No. : ";
            gets(nmobile);
            strcpy(mobile,nmobile);
            cout<<"\nRecords Updated Successfully...";
        }
        else
            cout<<"\nInvalid Input Provided...";
    }
}
//This function allow the ADMIN to make a new Record Entry for the
different new faculties...new faculty records are updated with the help of this
function
void getfadata()
{
    cout<<"\nEnter the Faculty Name : ";
    gets(name);
    cout<<"\nEnter the Faculty ID/Roll no. : ";
    cin>>rollno;
    cout<<"\nEnter the Faculty Father's Name : ";
    gets(fname);
    cout<<"\nEnter the Mobile No. : ";
    gets(mobile);
    cout<<"\nEnter the E-Mail Id : ";
    gets(mail);
    cout<<"\nEnter the Total Subjects : ";
    cin>>totsub;
    //cout<<"\nThe total subjects choosen are : "<<totsub;
    for(int i=0;i<totsub;i++)
    {

```

(Contd.)


```

        cout<<"\nEnter the Subject "<<i+1<<" Name : ";
        cin>>subject[i];
    }
    cout<<"\nCreate Your Login Password : ";
    cin>>passwd;
    cout<<"\nEnter the Unique Keyword to Recover Password : ";
    cin>>rpasswd;
    cout<<"\n\nPlease note your UserName is ID/Roll no.\n";
}
//This function displays all the relevant information to the admin related
to the Students at the ADMIN portals
void stdisplay()
{
    cout<<"\nStudent Name          : "<<name;
    cout<<"\nStudent ID/Roll No.    : "<<rollno;
    cout<<"\nStudent Father's Name  : "<<fname;
    cout<<"\nStudent Mobile No.     : "<<mobile;
    cout<<"\nStudent E-Mail ID      : "<<mail;
    cout<<"\nStudent Subjects      : "<<totsub;
    for(int i=0;i<totsub;i++)
    {
        cout<<"\n    Subject "<<i+1<<" : "<<subject[i];
    }
    if(totsub==0)
        cout<<"( None Subjects Specified... )";
}
//This function displays all the relevant information to the admin related
to the Faculty at the ADMIN portals
void fadisplay()
{
    cout<<"\nFaculty Name          : "<<name;
    cout<<"\nFaculty ID/Roll No.    : "<<rollno;
    cout<<"\nFaculty Father's Name  : "<<fname;
    cout<<"\nFaculty Mobile No.     : "<<mobile;
    cout<<"\nFaculty E-Mail ID      : "<<mail;
    cout<<"\nFaculty Subjects      : "<<totsub;
    for(int i=0;i<totsub;i++)
    {
        cout<<"\n    Subject "<<i+1<<" : "<<subject[i];
    }
    if(totsub==0)
        cout<<"None Subjects Specified...";
}
}a;
Admin f;
// a & f are the objects of the class Admin
void main()
{
    int ch;
    clrscr();
    cout<<"\n\n\n\t\tWelcome to RKGIT Database Portal ";
    cout<<"\n\n\n\t\t\tEnter to Continue ";
    getch();
    clrscr();

```

(Contd.)

```

cout<<"\n\n\n\t\t\tPress 1 for Admin Portal";
cout<<"\n\t\t\t\t\tPress 2 for Faculty Portal";
cout<<"\n\t\t\t\t\tPress 3 for Student Portal";
cout<<"\n\n\t\t\t\t\tEnter Your Choice : ";
cin>>ch;
clrscr();
if(ch==1)
{
    char adminuser[20],adminpass[20];
    cout<<"\n\t\t\t\t\tWelcome To Admin Login Portal ";
    cout<<"\n\nEnter the UserName : ";
    cin>>adminuser;
    cout<<"\nEnter the Password : ";
    for(int k=0;k<8;k++)
    {
        adminpass[k]=getch();
        cout<<"*";
    }
    getch();
    adminpass[k]=NULL;
    if((strcmp(adminuser,"admin")==0)&&(strcmp(adminpass,
    "password")==0))
    {
        clrscr();
    }
    else
    {
        cout<<"\n\n\t\t\t\t\t Invalid Access to Portal ";
        cout<<"\n\n\t\t\t\t\tThank You !!!";
        getch();
        exit(0);
    }
    char opera='y';
    do
    {
        int tmp;
        cout<<"\n\t\t\t\t\tWelcome To Admin Panel";
        cout<<"\n\nPress 1 to Add a Faculty Record ";
        cout<<"\nPress 2 to Add Multiple Records of Faculty ";
        cout<<"\nPress 3 to View All Records of Faculty ";
        cout<<"\nPress 4 to Delete a faculty Record ";
        cout<<"\nPress 5 to Add a Student Record ";
        cout<<"\nPress 6 to Add Multiple Records of Students ";
        cout<<"\nPress 7 to View All Records of Students ";
        cout<<"\nPress 8 to Delete a Student Record ";
        cout<<"\nPress 9 to Exit.";
        cout<<"\n\n\t\t\t\t\tEnter Your Choice : ";
        cin>>tmp;
        clrscr();
        if(tmp==1)//for inserting d single faculty records
        {
            cout<<"\nEnter the Details :- ";
            fstream fs;
            fs.open("fainfo.txt",ios::in|ios::out|ios::ate);

```

(Contd.)

```

        a.getfadata();
        fs.write((char *)&a,sizeof(Admin));
        fs.close();
        cout<<"\nRecord Entered Successfully...";
    }//for inserting d single faculty records
    if(tmp==2)//for inserting d multiple faculty records
    {
        int m=0;
        fstream fs;
        fs.open("fainfo.txt",ios::in|ios::out|ios::ate);
        do
        {
            cout<<"\nEnter the Details :- ";
            a.getfadata();
            fs.write((char *)&a,sizeof(Admin));
            cout<<
                "Press 0 if you want to Enter More Records : ";
            cin>>m;
        }while(m==0);
        fs.close();
        cout<<"\nRecord Entered Successfully...";
    }//for inserting d multiple faculty records
    if(tmp==3)//for view all faculty records
    {
        fstream fs;
        fs.open("fainfo.txt",ios::in);
        fs.seekg(0);
        while(!fs.eof())
        {
            fs.read((char *)&a,sizeof(Admin));
            a.fadisplay();
        }
        fs.close();
    }//for view all faculty records
    if(tmp==4)//for deleting a faculty record
    {
        char tmpfaid[15];
        int de=0,result=-1;
        cout<<"\nEnter the Faculty ID/Rollno. :- ";
        cin>>tmpfaid;
        fstream fs;
        fs.open("fainfo.txt",ios::in);
        fstream fs1;
        fs1.open("fanewinfo.txt",ios::out|ios::ate);
        while(!fs.eof())
        {
            fs.read((char *)&a,sizeof(Admin));
            result=strcmp(tmpfaid,a.rollno);
            if(result==0)
            {
                de=1;
            }
            else
                fs1.write((char *)&a,sizeof(Admin));
        }
    }
}

```

(Contd.)

```

    }
    if(de==1)
    {
        cout<<"\nRecord Deleted Successfully....";
    }
    else
    {
        cout<<"\nRecord not found....";
    }
    fs.close();
    fs1.close();
    remove("fainfo.txt") ;
    rename("fanewinfo.txt","fainfo.txt");
} //for deleting a faculty record
if(tmp==5) //for single student record input
{
    cout<<"\nEnter the Details :- ";
    fstream fs;
    fs.open
    ("stinfo.txt",ios::in|ios::out|ios::ate);
    a.getstdata();
    fs.write((char *)&a,sizeof(Admin));
    fs.close();
    cout<<"\nRecord Entered Successfully...";
} //for single student record input
if(tmp==6) //for multiple student record input
{
    int m=0;
    fstream fs;
    fs.open
    ("stinfo.txt",ios::in|ios::out|ios::ate);
    do
    {
        cout<<"\nEnter the Details :- ";
        a.getstdata();
        fs.write((char *)&a,sizeof(Admin));
        cout<<"Press 0 if you want to Enter More Records : ";
        cin>>m;
    } while(m==0);
    fs.close();
    cout<<"\nRecord Entered Successfully...";
} //for multiple student record input
if(tmp==7) //for view of all student record
{
    fstream fs;
    fs.open("stinfo.txt",ios::in);
    fs.seekg(0);
    while(!fs.eof())
    {
        fs.read((char *)&a,sizeof(Admin));
        a.stdisplay();
    }
    fs.close();
} //for view of all student record

```

(Contd.)

```

if(tmp==8)//for deleting a student record
{
    char tmpstid[15];
    int de=0,result=-1;
    cout<<"\nEnter the Student ID/Rollno. :- ";
    cin>>tmpstid;
    fstream fs;
    fs.open("stinfo.txt",ios::in);
    fstream fs1;
    fs1.open("stnewinfo.txt",ios::out|ios::ate);
    while(!fs.eof())
    {
        fs.read((char *)&a,sizeof(Admin));
        result=strcmp(tmpstid,a.rollno);
        if(result==0)
        {
            de=1;
        }
        else
            fs1.write((char *)&a,sizeof(Admin));
    }
    if(de==1)
    {
        cout<<"\nRecord Deleted Successfully....";
    }
    else
    {
        cout<<"\nRecord not found....";
    }
    fs.close();
    fs1.close();
    remove("stinfo.txt");
    rename("stnewinfo.txt","st info.txt");
}
//for deleting a student record
if(tmp==9)
{
    cout<<"\n\n\n\n\t\t\t\tThank You !!!";
    getch();
    exit(0);
}
//for exit
if(tmp<1||tmp>9)
{
    clrscr();
    cout<<"\n\n\n\t\t\t\tInvalid Input ....";
}
//for invalid input among choice of operation
getch();
cout<<"\n\nPress y ; For More Operations otherwise n : ";
cin>>opera;
if(opera!='y')
{
    getch();
    clrscr();
    cout<<"\n\n\n\n\t\t\t\tThank You !!!";
    getch();
}

```

(Contd.)

```

        } //closing of thank you
        } while (opera == 'Y' || opera == 'y');
    } //closing of admin login

    if (ch == 2) //begin of faculty view portal
    {
        clrscr();
        char un[20];
        int val, s = 0;
        cout << "\n\t\tWelcome to Faculty Login Page";
        cout << "\n\nEnter the UserName : ";
        cin >> un;
        fstream fs;
        fs.open("fainfo.txt", ios::in | ios::binary);
        fs.seekg(0);
        while (!fs.eof())
        {
            val = -1;
            fs.read((char *)&f, sizeof(Admin));
            val = f.login();
            if (val == 1)
            {
                s = 1;
                break;
            }
        }
        fs.close();
        if (s == 1)
        {
            clrscr();
        } //if first login is valid
        if (s != 1) //if first login is invalid then
        {
            clrscr();
            int ho = 0;
            cout << "\n\n\t\tYour Login Credentials are In-Correct";
            cout << "\n\nThe UserName is Your ID/Rollno.";
            cout << "\n\nThe Password is Case-Sensitive.";
            cout << "\n\nPress 1 to Re-Cover Password & 2 to Re-Attempt Login ";
            cout << "\n\nEnter the Choice : ";
            cin >> ho;
            if (ho == 1) //recover password
            {
                cout << "Enter the UserName :- ";
                cin >> un;
                fstream fs;
                fs.open
                ("fainfo.txt", ios::in | ios::binary);
                fs.seekg(0);
                int re, su = -1;
                while (!fs.eof())
                {
                    re = -1;

```

(Contd.)

```

        fs.read((char *)&f,sizeof(Admin));
        re=f.recover();
        if(re==1)
        {
            su=1;
            break;
        }
    }
    fs.close();
    if(su==1)
    {
        getch();
        clrscr();
        cout<<"\n\n\n\t\t\tThank You !!! ";
        getch();
        exit(0);
    }
    else
    {
        cout<<"\nYou are a Invalid User.";
        getch();
        exit(0);
    }
} //recover password
if(ho==2) //re-attempt of login
{

    cout<<"\n\nEnter the UserName : ";
    cin>>:un;
    fstream fs;
    fs.open
    ("fainfo.txt",ios::in|ios::binary);
    fs.seekg(0);
    int suc=-1,valu; //valu for storing login() returned value
    suc for success login
    while(!fs.eof())
    {
        valu=-1;
        fs.read((char *)&f,sizeof(Admin));
        valu=f.login();
        if(valu==1)
        {
            suc=1;
            break;
        }
    }
    fs.close();
    if(suc==1)
    {
        clrscr();
    }
    else
    {

```

(Contd.)

```

        getch();
        cout<<"\nYou are an Invalid User...";
        cout<<"\nThank You !!!";
        getch();
        exit(0);
        exit(0);
    }
}
if(ho!=1&&ho!=2)
{
    cout<<"\n\nInvalid Input Provided. ";
    cout<<"\n\n\t\t\tThank You !!!";
    getch();
    exit(0);
}
} //CLOSING OF first invalid login( forget password & recover password)
//Begin of Faculty
char con='y';
do
{
    clrscr();
    cout<<"\n\n\t\t\tWelcome to Faculty Panel  ";
    cout<<"\n\n\t\t\t\t\tYour UserId is : "<<::un;
    cout<<"\n\nPress 1 to View Your Profile.";
    cout<<"\nPress 2 to Know Your Subjects.";
    cout<<"\nPress 3 to Add a Subject.";
    cout<<"\nPress 4 to Delete a Subject.";
    cout<<"\nPress 5 to Modify Your Profile.";
    int choice;
    cout<<"\n\nEnter Your Choice : ";
    cin>>choice;
    if(choice==1)
    {
        fstream fs;
        fs.open("fainfo.txt",ios::in);
        fs.seekg(0);
        int x;
        while(!fs.eof())
        {
            x=0;
            fs.read((char *)&f,sizeof(Admin));
            x=f.faprofile();
            if(x==1)
            {
                break;
            }
        }
        fs.close();

    } //closing of choice = 1
    if(choice==2)
    {
        fstream fs;

```

(Contd.)


```

fs.open("fainfo.txt",ios::in);
fs.seekg(0);
int y;
while(!fs.eof())
{
    y=0;
    fs.read((char *)&f,sizeof(Admin));
    int y=f.knowfasub();
    if(y==1)
    {
        break;
    }
}
fs.close();
} //closing of choice=2
if(choice==3)
{
    fstream fs;
    fstream fsl;
    fs.open("fainfo.txt",ios::in|ios::binary);
    fsl.open("tmpfainfo.txt",ios::out|ios::ate);
    fs.seekg(0);
    while(!fs.eof())
    {
        fs.read((char *)&f,sizeof(Admin));
        f.addfasub();
        fsl.write((char *)&f,sizeof(Admin));
    }
    fs.close();
    fsl.close();
    remove("fainfo.txt");
    rename("tmpfainfo.txt","fainfo.txt");
} //closing of choice=3
if(choice==4)
{
    fstream fs;
    fs.open("fainfo.txt",ios::in|ios::binary);
    fstream fsl;
    fsl.open("delfainfo.txt",ios::out|ios::ate);
    fs.seekg(0);
    while(!fs.eof())
    {
        fs.read((char *)&f,sizeof(Admin));
        f.delfasub();
        fsl.write((char *)&f,sizeof(Admin));
    }
    fs.close();
    fsl.close();
    remove("fainfo.txt");
    rename("delfainfo.txt","fainfo.txt");
}
if(choice==5)
{

```

(Contd.)

```

        fstream fs;
        fstream fs1;
        fs.open("fainfo.txt",ios::in|ios::binary);
        fs.seekg(0);
        fs1.open("modfainfo.txt",ios::out|ios::ate);
        while(!fs.eof())
        {
            fs.read((char *)&f,sizeof(Admin));
            f.modfaprofile();
            fs1.write((char *)&f,sizeof(Admin));
        }
        fs.close();
        fs1.close();
        remove("fainfo.txt");
        rename("modfainfo.txt","fainfo.txt");

    }
    if(choice<1||choice>5)
        cout<<"\nInvalid Input Provided !!! ";

    cout<<"\n\n\t\t\t\tEnter To Continue";
    getch();
    cout<<"\n\nPress y to Continue ; otherwise n : ";
    cin>>con;
    if(con!='y' && con!='Y')
    {
        clrscr();
        cout<<"\n\n\n\n\n\n\t\t\t\tThank You !!! ";
        getch();
        exit(0);
    }
    }while(con=='y' || con=='Y');

} //close of faculty view

//begin of student view
if(ch==3) //begin of student view
{
    clrscr();
    //char un[20];
    int value,s1=0;
    cout<<"\n\t\t\t\tWelcome to Student Login Page";
    cout<<"\n\nEnter the UserName : ";
    cin>>:un;
    fstream fs;
    fs.open("stinfo.txt",ios::in|ios::binary);
    fs.seekg(0);
    while(!fs.eof())
    {
        value=-1;
        fs.read((char *)&f,sizeof(Admin));
        value=f.login();
        if(value==1)

```

(Contd.)

```

        {
            s1=1;
            break;
        }
    }
    fs.close();
    if(s1==1)
    {
        clrscr();
        //cout<<"\n\n\n\t\t\tWelcome to Student Page ";
    }//if first login is valid
    if(s1!=1)//if first login is invalid then
    {
        clrscr();
        int sho=0;
        cout<<"\n\n\t\tYour Login Credentials are In-Correct";
        cout<<"\nThe UserName is Your ID/Rollno.";
        cout<<"\nThe Password is Case-Sensitive.";
        cout<<"\nPress 1 to Re-Cover Password & 2 to Re-Attempt Login ";
        cout<<"\nEnter the Choice : ";
        cin>>sho;
        if(sho==1) //recover password
        {
            cout<<"Enter the UserName :- ";
            cin>>:un;
            fstream fs;
            fs.open
            ("stinfo.txt",ios::in|ios::binary);
            fs.seekg(0);
            int re,su=-1;
            while(!fs.eof())
            {
                re=-1;
                fs.read((char *)&f,sizeof(Admin));
                re=f.recover();
                if(re==1)
                {
                    su=1;
                    break;
                }
            }
            fs.close();
            if(su==1)
            {
                getch();
                clrscr();
                cout<<"\n\n\n\n\t\t\tThank You !!! ";
                getch();
                exit(0);
            }
            else
            {
                cout<<"\nYou are a Invalid User.";
            }
        }
    }
}

```

(Contd.)

```

        cout<<"\nThank You !!! ";
        getch();
        exit(0);
    }
} //recover password
if(sho==2) //re-attempt of login
{

    cout<<"\n\nEnter the UserName : ";
    cin>>:un;
    fstream fs;
    fs.open("stinfo.txt",ios::in|ios::binary);
    fs.seekg(0);
    int suc=-1,valu; //valu for storing login() returnd value
    suc for success login
    while(!fs.eof())
    {
        valu=-1;
        fs.read((char *)&f,sizeof(Admin));
        valu=f.login();
        if(valu==1)
        {
            suc=1;
            break;
        }
    }
    fs.close();
    if(suc==1)
    {
        clrscr();
    }
    else
    {
        getch();
        cout<<"\nYou are an Invalid User...";
        cout<<"\nThank You !!!";
        getch();
        exit(0);
        exit(0);
    }
}
if(sho!=1&&sho!=2)
{
    cout<<"\n\nInvalid Input Provided. ";
    cout<<"\n\n\t\t\tThank You !!!";
    getch();
    exit(0);
}
} //CLOSING OF first invalid login( forget password & recover password)
//getch();

//Begin of Student Panel

```

(Contd.)

```
char moreop='y';

do
{

clrscr();
cout<<"\n\n\t\t\tWelcome to Student Panel  ";
cout<<"\n\n\t\t\t\t\tYour UserId is : "<<::un;
cout<<"\n\nPress 1 to View Your Profile.";
cout<<"\nPress 2 to Know Your Subjects.";
cout<<"\nPress 3 to Add a Subject.";
cout<<"\nPress 4 to Delete a Subject.";
cout<<"\nPress 5 to Modify Your Profile.";
int inchoice;
cout<<"\n\nEnter Your Choice : ";
cin>>inchoice;

if(inchoice==1)
{
    fstream fs;
    fs.open("stinfo.txt",ios::in);
    fs.seekg(0);
    int x;
    while(!fs.eof())
    {
        x=0;
        fs.read((char *)&f,sizeof(Admin));
        x=f.stprofile();
        if(x==1)
        {
            break;
        }
    }
    fs.close();

}

} //closing of inchoice = 1
if(inchoice==2)
{
    fstream fs;
    fs.open("stinfo.txt",ios::in);
    fs.seekg(0);
    int y;
    while(!fs.eof())
    {
        y=0;
        fs.read((char *)&f,sizeof(Admin));
        int y=f.knowstsub();
        if(y==1)
        {
            break;
        }
    }
    fs.close();
}
```

(Contd.)

```

} //closing of choice=2
if(inchoice==3)
{
    fstream fs;
    fstream fs1;
    fs.open("stinfo.txt", ios::in | ios::binary);
    fs1.open("tmpstinfo.txt", ios::out | ios::ate);
    fs.seekg(0);
    while(!fs.eof())
    {
        fs.read((char *)&f, sizeof(Admin));
        f.addstsub();
        fs1.write((char *)&f, sizeof(Admin));
    }
    fs.close();
    fs1.close();
    remove("stinfo.txt");
    rename("tmpstinfo.txt", "stinfo.txt");
} //closing of choice=3
if(inchoice==4)
{
    fstream fs;
    fs.open("stinfo.txt", ios::in | ios::binary);
    fstream fs1;
    fs1.open("delstinfo.txt", ios::out | ios::ate);
    fs.seekg(0);
    while(!fs.eof())
    {
        fs.read((char *)&f, sizeof(Admin));
        f.delstsub();
        fs1.write((char *)&f, sizeof(Admin));
    }
    fs.close();
    fs1.close();
    remove("stinfo.txt");
    rename("delstinfo.txt", "stinfo.txt");
}
if(inchoice==5)
{
    fstream fs;
    fstream fs1;
    fs.open("stinfo.txt", ios::in | ios::binary);
    fs.seekg(0);
    fs1.open("modstinfo.txt", ios::out | ios::ate);
    while(!fs.eof())
    {
        fs.read((char *)&f, sizeof(Admin));
        f.modstprofile();
        fs1.write((char *)&f, sizeof(Admin));
    }
    fs.close();
    fs1.close();
    remove("stinfo.txt");
}

```

```

        rename("modstinfo.txt","stinfo.txt");
    }
    if(inchoice<1||inchoice>5)
    cout<<"\nInvalid Input Provided...";
    cout<<"\n\n\t\t\tEnter to Continue ";
    getch();
    cout<<"\n\nPress y , otherwise n to Perform More Operations : ";
    cin>>moreop;
    if(moreop!='Y'&&moreop!='y')
    {
        clrscr();
        cout<<"\n\n\n\t\t\tThank You !!!";
        getch();
        exit(0);
    }
    }while(moreop=='Y' || moreop=='y');
    getch();

}

//closing of student panel ch=3
if(ch<1||ch>3)
{
    cout<<"\nInvalid Input Provided !!! ";
    getch();
    clrscr();
    cout<<"\n\n\n\t\t\tThank You";
}
}
}

```

Application Output

The following is a series of screenshots depicting how the **CMS** application functions.

Accessing Admin Portal

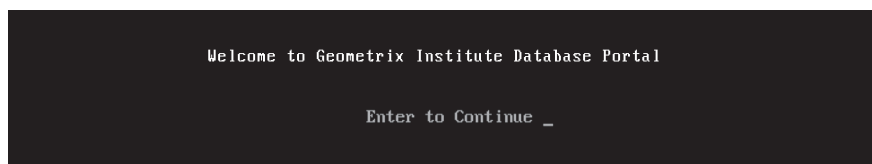


Fig. A.1

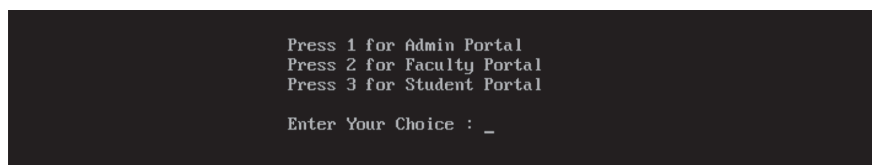


Fig. A.2

```

Press 1 for Admin Portal
Press 2 for Faculty Portal
Press 3 for Student Portal

Enter Your Choice : 1_

```

Fig. A.2.1

```

Welcome To Admin Login Portal

Enter the UserName : admin

Enter the Password : *****_

```

Fig. A.2.1.1

```

Welcome To Admin Panel

Press 1 to Add a Faculty Record
Press 2 to Add Multiple Records of Faculty
Press 3 to View All Records of Faculty
Press 4 to Delete a faculty Record
Press 5 to Add a Student Record
Press 6 to Add Multiple Records of Students
Press 7 to View All Records of Students
Press 8 to Delete a Student Record
Press 9 to Exit.

Enter Your Choice : _

```

Fig. A.2.1.2

```

Create Your Login Password : pankaj
Enter the Unique Keyword to Recover Password : pankaj

Please note your UserName is ID/Roll no.
Record Entered Successfully...

Press y : For More Operations otherwise n : y

Welcome To Admin Panel

Press 1 to Add a Faculty Record
Press 2 to Add Multiple Records of Faculty
Press 3 to View All Records of Faculty
Press 4 to Delete a faculty Record
Press 5 to Add a Student Record
Press 6 to Add Multiple Records of Students
Press 7 to View All Records of Students
Press 8 to Delete a Student Record
Press 9 to Exit.

Enter Your Choice :

```

Fig. A.2.1.4

```

Thank You !!!_

```

Fig. A.2.1.5

Resetting Password

```

Your Login Credentials are In-Correct
The UserName is Your ID/Rollno.
The Password is Case-Sensitive.
Press 1 to Re-Cover Password & 2 to Re-Attempt Login
Enter the Choice : 1
Enter the UserName :- 123456

Enter the Unique Keyword ( Provided by Admin ) : pankaj

You are a Valid user.
Your Password is pankaj
Please Exit To Login Again _

```

Fig. A.2.2.1.3

```

Your Login Credentials are In-Correct
The UserName is Your ID/Rollno.
The Password is Case-Sensitive.
Press 1 to Re-Cover Password & 2 to Re-Attempt Login
Enter the Choice : 2

Enter the UserName : 123456

Enter the Login Password : #####

```

Fig. A.2.2.1.4

Accessing Faculty Panel

```

Welcome to Faculty Panel

Your UserId is : 123456

Press 1 to View Your Profile.
Press 2 to Know Your Subjects.
Press 3 to Add a Subject.
Press 4 to Delete a Subject.
Press 5 to Modify Your Profile.

Enter Your Choice :

```

Fig. A.2.2.2

```

Welcome to Faculty Panel

Your UserId is : 123456

Press 1 to View Your Profile.
Press 2 to Know Your Subjects.
Press 3 to Add a Subject.
Press 4 to Delete a Subject.
Press 5 to Modify Your Profile.

Enter Your Choice : 2


Faculty Total Subjects :- 2
Subject 1 : ds
Subject 2 : os

Enter To Continue

Press y to Continue ; otherwise n : y

```

Fig. A.2.2.4



```
Thank You !!!_
```

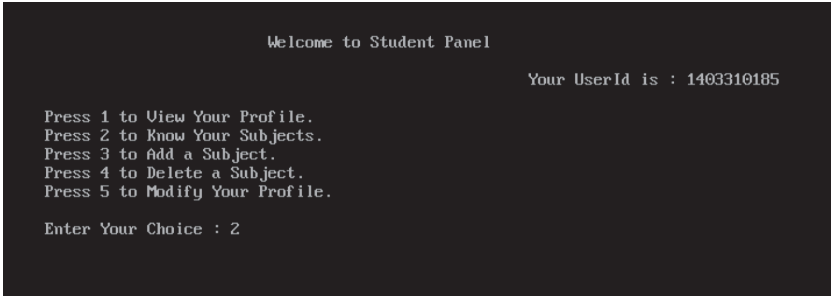
Fig. A.2.2.5

Accessing Student Panel



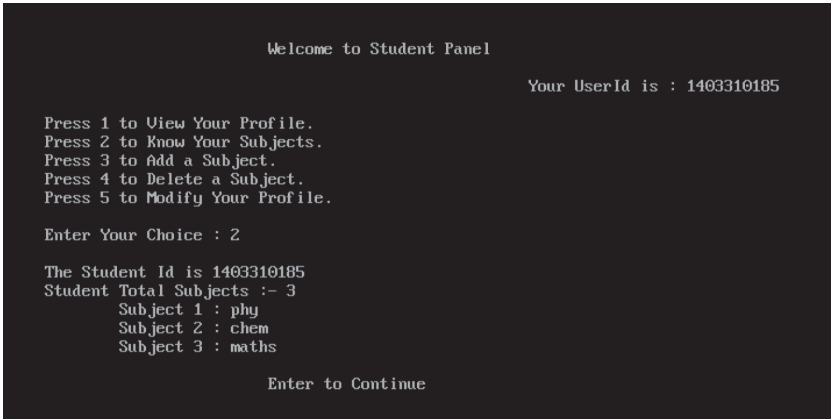
```
Welcome to Student Login Page  
Enter the UserName : 1403310185  
Enter the Login Password : #####
```

Fig. A.2.3.1



```
Welcome to Student Panel  
Your UserId is : 1403310185  
  
Press 1 to View Your Profile.  
Press 2 to Know Your Subjects.  
Press 3 to Add a Subject.  
Press 4 to Delete a Subject.  
Press 5 to Modify Your Profile.  
  
Enter Your Choice : 2
```

Fig. A.2.3.2



```
Welcome to Student Panel  
Your UserId is : 1403310185  
  
Press 1 to View Your Profile.  
Press 2 to Know Your Subjects.  
Press 3 to Add a Subject.  
Press 4 to Delete a Subject.  
Press 5 to Modify Your Profile.  
  
Enter Your Choice : 2  
  
The Student Id is 1403310185  
Student Total Subjects :- 3  
Subject 1 : phy  
Subject 2 : chem  
Subject 3 : maths  
  
Enter to Continue
```

Fig. A.2.3.3

```
Welcome to Student Panel

Your UserId is : 1403310185

Press 1 to View Your Profile.
Press 2 to Know Your Subjects.
Press 3 to Add a Subject.
Press 4 to Delete a Subject.
Press 5 to Modify Your Profile.

Enter Your Choice : 2

The Student Id is 1403310185
Student Total Subjects :- 3
    Subject 1 : phy
    Subject 2 : chem
    Subject 3 : maths

Enter to Continue

Press y , otherwise n to Perform More Operations : y_
```

Fig. A.2.3.4

```
Thank You !!!_
```

Fig. A.2.3.5

A.2 MAJOR PROJECT – PAY SLIP GENERATION SYSTEM

Learning Objectives

After going through this project, you will be able to

- ☐ Learn how real-world applications are developed using C++ programming
- ☐ Develop menu-driven applications
- ☐ Perform file handling operations such as read, write, append, and check for various I/O errors
- ☐ Represent the output in a formatted manner by using various I/O manipulators

Application Overview

The objective of **Pay Slip Generation System (PSG System)** is to store the details of the employees of an organization and generate their pay slips. The organization employs two types of employees, viz., permanent and contractual. The **permanent** employees are entitled to dearness allowance (DA), house rent allowance (HRA), medical allowance (MA), provident fund deductions (PF), professional tax deductions (PTax) and income tax deductions (ITax) on their basic salary.

The **contractual** employees are entitled to a lump sum gross salary with professional and income tax deductions. The **PSG system** stores and retrieves the details of both types of employees, and also computes and generates their monthly pay slips. The system is capable of handling varying DA, HRA rates along with changes to the medical allowance and professional tax (if any).

Key Elements

The class diagram for the PSG system application is shown below:

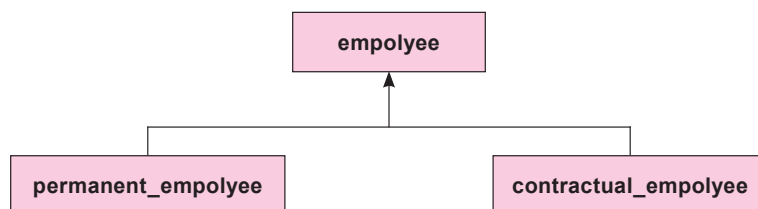


Table A.3 lists the various members of the employee class.

Table A.3 *employee Class Members*

employee class		
Name	Type	Description
emp_no	Attribute	Unique ID of the employees
emp_name	Attribute	Name of employee
emp_add	Attribute	Address of employee
emp_dept	Attribute	Department of employee
emp_desg	Attribute	Designation of employee
getdata	Member function	Accept data from user
displaydata	Member function	Displays data as output

Table A.4 *permanent_employee Class Members*

permanent_employee class		
Name	Type	Description
emp_type	Attribute	Permanent employee
emp_basic	Attribute	Basic salary of employee
emp_da	Attribute	Dearness allowance
emp_hra	Attribute	House rent allowance
emp_ma	Attribute	Medical allowance
emp_pf	Member function	Provident fund
ptax	Member function	Professional tax
itax	Attribute	Income tax
gross	Attribute	Gross salary
Net	Attribute	Net salary
getdata	Member function	Accepts data from user
displaydata	Member function	Displays data
calculate	Member function	Calculates DA, HRA, PF, gross and net

(Contd.)

Table A.3 (Contd.)

Name	Type	Description
search	Friend function	Searches for permanent employee
store_in_file	Friend function	Stores details of employee in file
Generate_pay_slip	Friend function	Generates pay slip and stores it in file
Compute_OT	Friend function	Computes the overtime dues

Table A.5 lists the various members of the contractual_employee class:

Table A.5 contractual_employee Class Members

contractual_employee class		
Name	Type	Description
emp_type	Attribute	Contractual employee
ptax	Member function	Professional tax
itax	Attribute	Income tax
gross	Attribute	Gross salary
net	Attribute	Net salary
getdata	Member function	Accepts data from user
displaydata	Member function	Displays data
calculate	Member function	Calculates net by deducting taxes
search	Friend function	Searches for contractual employee
store_in_file	Friend function	Stores details of employee in file
Generate_pay_slip	Friend function	Generates pay slip and stores it in file
Compute_OT	Friend function	Computes the overtime dues

Program Logic

Salary Calculation for Permanent Employees

The gross salary of every permanent employee has the following components:

1. Basic
2. Dearness allowance: Calculated as a pre-set percentage of basic
3. House rent allowance: Calculated as a pre-set percentage of basic
4. Medical allowance: A fixed pre-set lump sum amount

The net salary is computed by subtracting from the gross the following:

- Provident fund: Calculated as 12% of (Basic + DA)
- Professional tax: Fixed amount
- Income tax: Fixed amount (The tax computations are usually finalized at the end of each financial year)

Salary Calculation for Permanent Employees

Every contractual employee is entitled to receive a fixed gross salary.

The net salary is computed by subtracting from the gross the following:

- Professional tax: Fixed amount
- Income tax: Fixed amount (The tax computations are usually finalized at the end of every financial year)

Overtime Dues Calculations for all Employees

The employees are entitled to receive overtime allowance which has been fixed at the rate of INR 400 per hour. The overtime is thus, calculated in batch mode after accepting the number of overtime hours for each employee as input from the user of the system.

Application Code

The following is the code for the PSG system. This code has been developed in the Linux environment. Thus, some commonly used functions such as `getch()`, `clrscr()`, etc. have not been used. The `getline()` function calls may require the buffer to be flushed prior to being called.

PSG SYSTEM APPLICATION

```
#include<iostream>
#include <fstream>
#include <cstdlib>
#include <cstring>

using namespace std;

//forward declaration of the classes
class pay_slip;
class permanent_employee;
class contractual_employee;

//function prototypes
int get_da_rate();
int get_hra_rate();
int get_ma();
int get_ptax();
int get_emp_no();
int search(int, permanent_employee *);
int search(int, contractual_employee *);
void generate_pay_slip(permanent_employee);
void generate_pay_slip(contractual_employee);
void compute_OT();

class employee{//base class
protected:
    int emp_no;
```

(Contd.)

```

        char emp_name[40];
        char emp_add[80];
        char emp_desg[20];
        char emp_dept[20];
    public:
        void getdata();
        void displaydata();
}; //end of base class definitions

void employee::getdata() {
    emp_no = get_emp_no();
    cout<<"\nEnter the name of the employee:";
    cin.getline(emp_name, 40);
    cout<<"\nEnter the address of the employee:";
    cin.getline(emp_add, 80);
    cout<<"\nEnter the designation of the employee:";
    cin.getline(emp_desg, 20);
    cout<<"\nEnter the department of the employee:";
    cin.getline(emp_dept, 20);
}

void employee::displaydata() {
    cout<<"\nEmp. No.: "<<emp_no;
    cout<<"\nName: "<<emp_name;
    cout<<"\nAddress: "<<emp_add;
    cout<<"\nDesignation: "<<emp_desg;
    cout<<"\nDepartment: "<<emp_dept;
}

class permanent_employee: public employee{//derived from base class
private:
    char emp_type[10];
    long emp_basic;
    long emp_da;
    long emp_hra;
    int emp_ma;
    long emp_pf;
    int ptax;
    int itax;
    long gross;
    long net;
public:
    void getdata();//over-ridden function
    void displaydata();//over-ridden function
    void calculate();
    void store_in_file();
    friend int search(int, permanent_employee *);
    friend void generate_pay_slip(permanent_employee);
    friend void compute_OT();
};

void permanent_employee::getdata() {
    employee::getdata();//calling base getdata()
    try {//exception handling for negative basic

```

(Contd.)

```

        cout<<"\nEnter the basic salary: ";
        cin>>emp_basic;
        if(emp_basic<0)
            throw emp_basic;
    }
    catch(...) {
        cout<<"\nBasic should be positive";
        return;
    }
    try { //exception handling for negative income tax
        cout<<"\nEnter the income tax to be deducted: ";
        cin>>itax;
        if(itax<0)
            throw itax;
    }
    catch(...) {
        cout<<"\nIncome Tax should be positive";
        return;
    }
    calculate();
}

void permanent_employee::calculate(){
    int da, hra;

    strcpy(emp_type, "Permanent");
    da = get_da_rate();
    emp_da = emp_basic * da / 100; //da is calculated as
                                   // a percentage of basic
    hra = get_hra_rate();
    emp_hra = emp_basic * hra / 100; //hra is calculated as
                                   //a percentage of basic
    emp_ma = get_ma();
    emp_pf = (emp_basic + emp_da) * 12 / 100; //pf is calculated as a
                                   //percentage of basic+da
    ptax = get_ptax();
    gross = emp_basic + emp_da + emp_hra + emp_ma; //calc. of gross
    net = gross - ptax - itax - emp_pf; //calc of net
}

void permanent_employee::displaydata(){
    employee::displaydata(); //calling base displaydata()
    cout<<"\nEmployee type: "<<emp_type;
    cout<<"\nBasic: "<<emp_basic;
    cout<<"\nDearness Allowance: "<<emp_da;
    cout<<"\nHRA: "<<emp_hra;
    cout<<"\nMedical Allowance: "<<emp_ma;
    cout<<"\nPF: "<<emp_pf;
    cout<<"\nProfessional tax: "<<ptax;
    cout<<"\nIncome tax: "<<itax;
    cout<<"\nGross Salary: "<<gross;
    cout<<"\nNet Salary: "<<net;
}

```

(Contd.)


```

void permanent_employee::store_in_file(){
    ofstream outfile;
    outfile.open("per_empf", ios::app); //Open per_empf in append mode
    //file per_empf stores details of permanent employees
    outfile<<emp_no<<"\n";
    outfile<<emp_name<<"\n"<<emp_add<<"\n"<<emp_desg<<"\n";
    outfile<<emp_dept<<"\n"<<emp_type<<"\n"<<emp_basic<<"\n"<<
        emp_da<<"\n"<<emp_hra<<"\n"<<emp_ma<<"\n"<<emp_pf<<"\n"<<
        ptax<<"\n"<<itax<<"\n"<<gross<<"\n"<<net<<"\n";
    outfile.close(); //close file
}

class contractual_employee: public employee{//derived from base employee
private:
    char emp_type[12];
    long gross;
    int ptax;
    int itax;
    long net;
public:
    void getdata();
    void displaydata();
    void calculate();
    void store_in_file();
    friend int search(int, contractual_employee *);
    friend void generate_pay_slip(contractual_employee);
    friend void compute_OT();
};

void contractual_employee::getdata(){//over-ridden function getdata()
    employee::getdata();//calling base getdata()
    try{//exception for negative gross
        cout<<"\nEnter the gross salary: ";
        cin>>gross;
        if(gross<0)
            throw gross;
    }
    catch(...){
        cout<<"\nGross Salary should be positive";
        return;
    }
    try{//exception for negative income tax
        cout<<"\nEnter the income tax salary: ";
        cin>>itax;
        if(itax<0)
            throw itax;
    }
    catch(...){
        cout<<"\nGross Income Tax should be positive";
        return;
    }
    calculate();
}

```

(Contd.)

```

    }

    void contractual_employee::calculate() {
        strcpy(emp_type, "Contractual");
        ptax = get_ptax();
        net = gross - ptax - itax;
    }

    void contractual_employee::displaydata() { //over-ridden displaydata()
        employee::displaydata(); //calling base displaydata
        cout<<"\nEmployee Type: "<<emp_type;
        cout<<"\nGross Salary: "<<gross;
        cout<<"\nProfessional tax: "<<ptax;
        cout<<"\nIncome tax: "<<itax;
        cout<<"\nNet Salary: "<<net;
    }

    void contractual_employee::store_in_file() {
        ofstream outfile;
        outfile.open("con_empf", ios::app); //open file in append mode
        //file con_empf stores details of contractual employees
        outfile<<emp_no<<"\n";
        outfile<<emp_name<<"\n"<<emp_add<<"\n"<<emp_desg<<"\n";
        outfile<<emp_dept<<"\n"<<emp_type<<"\n"<<gross<<"\n"
            <<ptax<<"\n"<<itax<<"\n"<<net<<"\n";
        outfile.close(); //close files
    }

    int search(int no, permanent_employee *per) { //func. defn. for searching
        //permanent employees from file
        ifstream infile;
        permanent_employee p;

        char ch;
        infile.open("per_empf"); //open file in read mode
        if(!infile.fail()) //checking if file exists
            while(!infile.eof()) { //while not end of file
                infile>>p.emp_no;
                infile.ignore(1000, '\n');
                infile.getline(p.emp_name, 40);
                infile.getline(p.emp_add, 80);
                infile.getline(p.emp_desg, 20);
                infile.getline(p.emp_dept, 20);
                infile>>p.emp_type>>p.emp_basic>>p.emp_da>>p.emp_hra
                >>p.emp_ma>>p.emp_pf>>p.ptax>>p.itax>>p.gross>>p.net;
                if(no == p.emp_no) { //looks for a match
                    *per = p; //set pointer to object if found
                    return 1; //return true
                }
            }
        return 0; //return false if not found
    }

    int search (int no, contractual_employee *con) { //func. defn. for

```

(Contd.)

```

        //searching contractual employees from file
ifstream infile;
contractual_employee p;
infile.open("con_empf");//open file in read mode
if(!infile.fail())//checking if file exists
    while(!infile.eof()){//while not end of file
        infile>>p.emp_no;
        infile.ignore(1000, '\n');
        infile.getline(p.emp_name, 40);
        infile.getline(p.emp_add, 80);
        infile.getline(p.emp_desg, 20);
        infile.getline(p.emp_dept, 20);
        infile>>p.emp_type>>p.gross
            >>p.ptax>>p.itax>>p.net;
        if(no == p.emp_no){//looks for a match
            *con = p;//set pointer to object if found
            return 1;//return true
        }
    }
    return 0;//return false if not found
}

void generate_pay_slip(permanent_employee per){
//function to generate pay slips of permanent employees
int no;
char fl_name[8], num[8], emp[8], c;

cout<<"\n\nEnter month no (1..12): ";
cin>>no;
if(no>12 || no<0){
    cout<<"\n\nInvalid month number";
    return;
}
system("clear");
cout<<"::::::::::::Pay Slip for Month number "<<no<<"::::::::::::\n\n";
per.calculate();//perform calculation
per.displaydata();//display data about employee
//create file name to store pay slip of employee by
//concatenating employee no. with month no. with an
//underscore character in between
sprintf(num, "%d_", no);
sprintf(emp, "%d", per.emp_no);
strcpy(fl_name, num);
strcat(fl_name, emp);
fl_name[strlen(num)+strlen(emp)]='\0';
ofstream outfile(fl_name);
//store pay slip in file
outfile<<"::::::::::::Pay Slip for Month number "<<no<<"::::::::::::\n\n";
outfile<<"Emp. Number: "<<per.emp_no<<"\n";
outfile<<"Emp. Name: "<<per.emp_name<<"\n";
outfile<<"Emp. Add: "<<per.emp_add<<"\n";
outfile<<"Emp. Designation: "<<per.emp_desg<<"\n";
outfile<<"Emp. Department: "<<per.emp_dept<<"\n";
outfile<<"Emp. Type: "<<per.emp_type<<"\n"<<"Emp. Basic: "<<per.emp_

```

(Contd.)

```

basic<<"\n"
    "Dearness Allowance:  "<<per.emp_da<<"\n"<<"House Rent Allowance:  "<<per.
emp_hra
    <<"\n"<<"Medical Allowance:  "<<per.emp_ma<<"\n"<<"Provident Fund:  "<<per.
emp_pf<<"\n"
    <<"Professional tax:  "<<per.ptax<<"\n"<<"Income tax:  "<<per.itax<<"\n"
    <<"Gross Salary:  "<<per.gross<<"\n"<<"Net Salary:  "<<per.net<<"\n";
    outfile.close();
}

void generate_pay_slip(contractual_employee per){
//function to generate pay slips of contractual employees
    int no;
    char fl_name[8], num[8], emp[8], c;

    cout<<"\n\nEnter month no (1..12):  ";
    cin>>no;
    if(no>12 || no<0){
        cout<<"\n\nInvalid month number";
        return;
    }
    system("clear");

    per.calculate();//perform calculation
    per.displaydata();//display details of contractual employee
        //create file name to store pay slip of
        //contractual employee by concatenating
        //employee no. with month no. with an
        //underscore character in between
    sprintf(num, "%d_", no);
    sprintf(emp, "%d", per.emp_no);
    strcpy(fl_name, num);
    strcat(fl_name, emp);
    fl_name[strlen(num)+strlen(emp)]='\0';
    ofstream outfile(fl_name);
        //store pay slip in file
    outfile<<"::::::::::::Pay Slip for Month number "<<no<<"::::::::::::";
    outfile<<"Emp. Number:  "<<per.emp_no<<"\n";
    outfile<<"Emp. Name:  "<<per.emp_name<<"\n";
    outfile<<"Emp. Add:  "<<per.emp_add<<"\n";
    outfile<<"\nEmp. Designation:  "<<per.emp_desg<<"\n";
    outfile<<"Emp. Department:  "<<per.emp_dept<<"\n";
    outfile<<"Emp. Type:  "<<per.emp_type<<"\n"<<"Gross Salary:  "
        <<per.gross<<"\n"<<"Professional Tax:  "<<per.ptax<<"\n"
        <<"Income Tax:  "<<per.itax<<"\n"<<"Net Salary:  "<<per.net<<"\n";
    outfile.close();
}

void compute_OT(){//function to compute Overtime dues
    ifstream infile;
    permanent_employee p;
    contractual_employee c;
    int hours;
    ofstream outfile;

```

(Contd.)

```

infile.open("per_empf");
outfile.open("per_payroll");
system("clear");
cout<<"\n\nComputation of Overtime Dues for Permanent Employee in
progress.....";
while(!infile.eof()){
    infile>>p.emp_no;
    infile.ignore(1000, '\n');
    infile.getline(p.emp_name, 40);
    infile.getline(p.emp_add, 80);
    infile.getline(p.emp_desg, 20);
    infile.getline(p.emp_dept, 20);
    infile>>p.emp_type>>p.emp_basic>>p.emp_da>>p.emp_hra
        >>p.emp_ma>>p.emp_pf>>p.ptax>>p.itax>>p.gross>>p.net;
    if(infile.eof())//needed to check if eof flag has been set
        //else last employee details will be read twice
        break;
    cout<<"\nEnter the overtime hours for employee no. "<<p.emp_no<<" : ";
    cin>>hours;
    p.net=p.net+(hours*400);//Over time rate is 400 per hour
    outfile<<p.emp_no<<"\n";
    outfile<<p.emp_name<<"\n"<<p.emp_add<<"\n"<<p.emp_desg<<"\n";
    outfile<<p.emp_dept<<"\n"<<p.emp_type<<"\n"<<p.emp_basic<<"\n"<<
p.emp_da<<"\n"<<p.emp_hra<<"\n"<<p.emp_ma<<"\n"<<p.emp_pf<<"\n"<<
p.ptax<<"\n"<<p.itax<<"\n"<<p.gross<<"\n"<<hours*400<<"\n"<<p.net<<"\n";
}
infile.close();
outfile.close();
cout<<"\nOvertime Dues computation for permanent employees complete";
cin.get();
infile.open("con_empf");
outfile.open("con_payroll");
system("clear");
cout<<"\n\nComputation of Overtime Dues for Contractual Employee in
progress.....";
while(! (infile.eof())){
    infile>>c.emp_no;
    infile.ignore(1000, '\n');
    infile.getline(c.emp_name, 40);
    infile.getline(c.emp_add, 80);
    infile.getline(c.emp_desg, 20);
    infile.getline(c.emp_dept, 20);
    infile>>c.emp_type>>c.gross
        >>c.ptax>>c.itax>>c.net;
    if(infile.eof())
        break;
    cout<<"\nEnter the overtime hours for employee no. "<<c.emp_no<<" : ";
    cin>>hours;
    c.net=c.net+(hours*400);
    outfile<<c.emp_no<<"\n";
    outfile<<c.emp_name<<"\n"<<c.emp_add<<"\n"<<c.emp_desg<<"\n";
    outfile<<c.emp_dept<<"\n"<<c.emp_type<<"\n"<<c.gross<<"\n"
        <<c.ptax<<"\n"<<c.itax<<"\n"<<hours*400<<"\n"<<c.net<<"\n";
}

```

(Contd.)

```

infile.close();
outfile.close();
cout<<"\nOvertime Dues computation for contractual employees complete";
cin.get();
system("clear");
cout<<"\nThe Overtime Dues for the employees are as follows:\n\n";
infile.open("per_payroll");
cout.setf(ios::left, ios::adjustfield); //display output in
                                     //formatted fashion

cout.width(10);
cout<<"Emp No.";
cout.setf(ios::left, ios::adjustfield);
cout.width(40);
cout<<"Emp. Name";
cout.setf(ios::left, ios::adjustfield);
cout.width(10);
cout<<"Overtime Dues"<<"\n\n";
while(!infile.eof()){
    infile>>p.emp_no;
    infile.ignore(1000, '\n');
    infile.getline(p.emp_name, 40);
    infile.getline(p.emp_add, 80);
    infile.getline(p.emp_desg, 20);
    infile.getline(p.emp_dept, 20);
    infile>>p.emp_type>>p.emp_basic>>p.emp_da>>p.emp_hra
        >>p.emp_ma>>p.emp_pf>>p.ptax>>p.itax>>p.gross>>hours>>p.net;
    if(infile.eof())//needed to avoid duplication of last
        //employee details
        break;
    cout.setf(ios::left, ios::adjustfield);
    cout.width(10);
    cout<<p.emp_no;
    cout.setf(ios::left, ios::adjustfield);
    cout.width(40);
    cout<<p.emp_name;
    cout.width(10);
    cout<<hours<<"\n";
}
infile.close();
infile.open("con_payroll");
while(!(infile.eof())){
    infile>>c.emp_no;
    infile.ignore(1000, '\n');
    infile.getline(c.emp_name, 40);
    infile.getline(c.emp_add, 80);
    infile.getline(c.emp_desg, 20);
    infile.getline(c.emp_dept, 20);
    infile>>c.emp_type>>c.gross
        >>c.ptax>>c.itax>>hours>>c.net;
    if(infile.eof())
        break;
    cout.setf(ios::left, ios::adjustfield);

```

(Contd.)

```
        cout.width(10);
        cout<<c.emp_no;
        cout.setf(ios::left, ios::adjustfield);
        cout.width(40);
        cout<<c.emp_name;
        cout.width(10);
        cout<<hours<<"\n";
    }
    infile.close();
}

int get_da_rate(){//function to retrieve da rate from file
    int da;
    ifstream infile("da_file");
    infile>>da;
    infile.close();
    return da;
}

int get_hra_rate(){//function for retrieving hra rate from file
    int hra;
    ifstream infile("hra_file");
    infile>>hra;
    infile.close();
    return hra;
}

int get_ma(){//function to retrieve medical allowance from file
    int ma;
    ifstream infile("ma_file");
    infile>>ma;
    infile.close();
    return ma;
}

int get_ptax(){//function to retrieve professional tax from file
    int ptax;
    ifstream infile("ptax_fl");
    infile>>ptax;
    infile.close();
    return ptax;
}

void set_da_rate(int da){//function to set the da rate
    ofstream outfile("da_file");
    outfile<<da;
    outfile.close();
}

void set_hra_rate(int hra){//function to set the hra rate
    ofstream outfile("hra_file");
    outfile<<hra;
```

(Contd.)

```

        outfile.close();
    }

    void set_ma(int ma){//function to set the medical allowance
        ofstream outfile("ma_file");
        outfile<<ma;
        outfile.close();
    }

    void set_ptax(int ptax){//function to set the professional tax
        ofstream outfile("ptax_fl");
        outfile<<ptax;
        outfile.close();
    }

    int get_emp_no(){//function for auto-generating employee number
        ifstream infile;
        int no;
        infile.open("empno_fl");
        if(!infile.fail()){
            infile>>no;
            no = no+1;
            infile.close();
        }
        else no = 1;
        ofstream outfile("empno_fl");
        outfile<<no;
        outfile.close();
        return no;
    }

    int main(){
        int choice, emp_type, no;
        int da, hra, ma, ptax;
        permanent_employee per_person;
        contractual_employee con_person;

        while(1){
            system("clear");
            cout<<"\n:::::::::MENU:::::::::";
            cout<<"\n1. New Employee";
            cout<<"\n2. Generate Pay Slip";
            cout<<"\n3. Set Dearness Allowance rate";
            cout<<"\n4. Set HRA rate";
            cout<<"\n5. Set Professional tax";
            cout<<"\n6. Set Medical Allowance rate";
            cout<<"\n7. Display Detail of Employee";
            cout<<"\n8. Compute Overtime Dues for Employees";
            cout<<"\n9. Exit";
            cout<<"\n\nEnter your choice:: ";
            cin>>choice;
            fflush(stdin);

```

(Contd.)


```

system("clear");
switch(choice){
case 1://Entering new employee details
    cout<<"\nEnter the employee type";
    cout<<"\n1. Permanent Employee";
    cout<<"\n2. Contractual Employee";
    cin>>emp_type;
    switch(emp_type){
    case 1://permanent employee
        per_person.getdata();
        per_person.store_in_file();
        break;
    case 2://contractual employee
        con_person.getdata();
        con_person.store_in_file();
        break;
    default:
        cout<<"\n\nWrong type";
        break;
    }
    break;
case 2://generate payslip
    cout<<"\n\nEnter Employee no.: ";
    cin>>no;
    cout<<"\n\nEnter Employee type: ";
    cout<<"\n1. Permanent Employee";
    cout<<"\n2. Contractual Employee";
    cin>>emp_type;
    switch(emp_type){
    case 1://for permanent employee
        if((search(no, &per_person))==0){
            cout<<"\n\nInvalid Employee number";
        }
        else generate_pay_slip(per_person);
        break;
    case 2://for contractual employee
        if((search(no, &con_person))==0){
            cout<<"\n\nInvalid Employee number";
        }
        else generate_pay_slip(con_person);
        break;
    default:
        cout<<"\n\nWrong type";
        break;
    }
    break;
case 3://Set DA rate
    cout<<"\n\nEnter new Dearness Allowance rate: ";
    cin>>da;
    set_da_rate(da);
    break;
case 4://Set HRA rate

```

(Contd.)

```

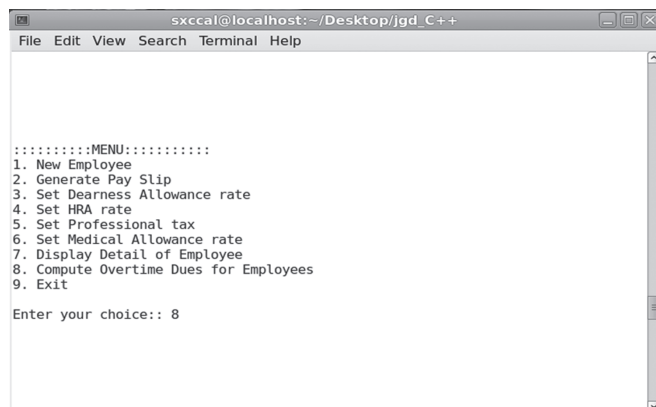
        cout<<"\n\nEnter new HRA rate:  ";
        cin>>hra;
        set_hra_rate(hra);
        break;
    case 5://Set Professional Tax
        cout<<"\n\nEnter new Professional Tax:  ";
        cin>>ptax;
        set_ptax(ptax);
        break;
    case 6://Set Medical Allowance
        cout<<"\n\nEnter new Medical Allowance:  ";
        cin>>ma;
        set_ma(ma);
        break;
    case 7://Search for Employee
        cout<<"\n\nEnter Employee no.:  ";
        cin>>no;
        cout<<"\n\nEnter Employee type:  ";
        cout<<"\n1. Permanent Employee";
        cout<<"\n2. Contractual Employee";
        cin>>emp_type;
        switch(emp_type){
            case 1://Permanent Employee
                if((search(no, &per_person))==0){
                    cout<<"\n\nInvalid Employee number";
                }
                else per_person.displaydata();
                break;
            case 2://Contractual employee
                if((search(no, &con_person))==0){
                    cout<<"\n\nInvalid Employee number";
                }
                else con_person.displaydata();
                break;
            default:
                cout<<"\n\nWrong type";
                break;
        }
        break;
    case 8://Compute Overtime Dues for Employees
        compute_OT();
        break;
    case 9://Exit System
        exit(0);
    default:
        cout<<"\n\nWrong choice";
    } //end of switch case for main menu
} //End of While loop
return 0;
} //End of Main function

```

Application Output

The following is a series of screenshots depicting how the PSG system functions.

Main Menu



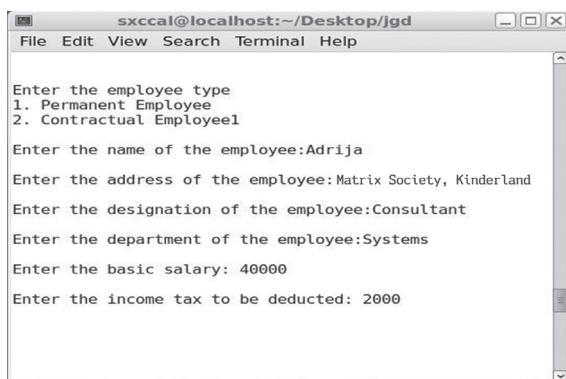
```

sxccal@localhost:~/Desktop/jgd_C++
File Edit View Search Terminal Help

:MENU:
1. New Employee
2. Generate Pay Slip
3. Set Dearness Allowance rate
4. Set HRA rate
5. Set Professional tax
6. Set Medical Allowance rate
7. Display Detail of Employee
8. Compute Overtime Dues for Employees
9. Exit

Enter your choice:: 8
  
```

Entering details of permanent employee (Option 1 of main menu)



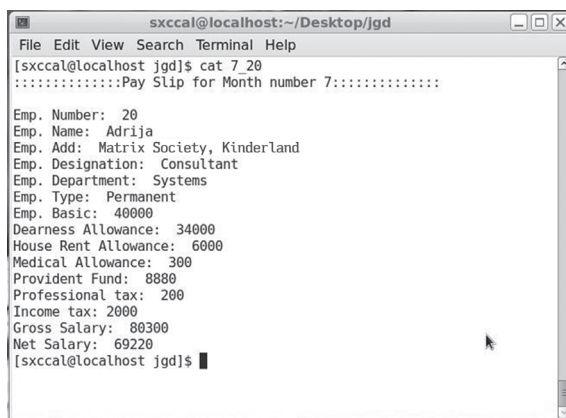
```

sxccal@localhost:~/Desktop/jgd
File Edit View Search Terminal Help

Enter the employee type
1. Permanent Employee
2. Contractual Employee1

Enter the name of the employee:Adrija
Enter the address of the employee:Matrix Society, Kinderland
Enter the designation of the employee:Consultant
Enter the department of the employee:Systems
Enter the basic salary: 40000
Enter the income tax to be deducted: 2000
  
```

Pay slip generated and stored for permanent employee (Option 2 of main menu)

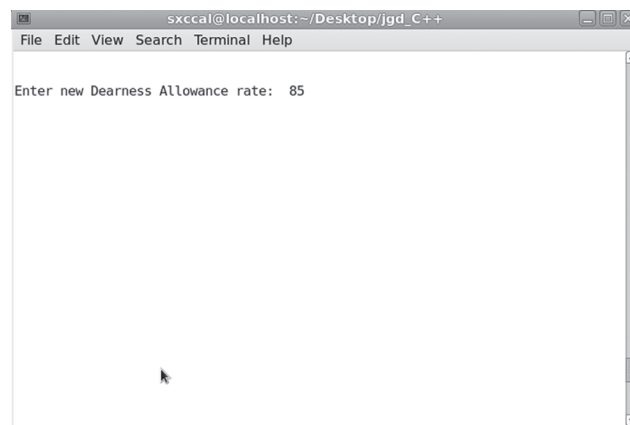


```

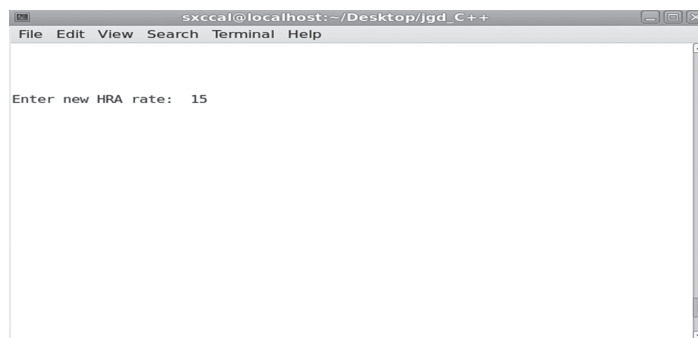
sxccal@localhost:~/Desktop/jgd
File Edit View Search Terminal Help

[sxccal@localhost jgd]$ cat 7_20
:Pay Slip for Month number 7:
Emp. Number: 20
Emp. Name: Adrija
Emp. Add: Matrix Society, Kinderland
Emp. Designation: Consultant
Emp. Department: Systems
Emp. Type: Permanent
Emp. Basic: 40000
Dearness Allowance: 34000
House Rent Allowance: 6000
Medical Allowance: 300
Provident Fund: 8800
Professional tax: 200
Income tax: 2000
Gross Salary: 80300
Net Salary: 69220
[sxccal@localhost jgd]$
  
```

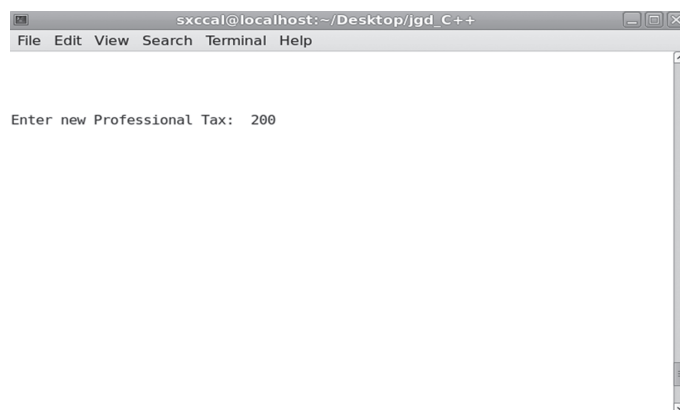
Setting the Dearness Allowance Rate (Option 3 of main menu)



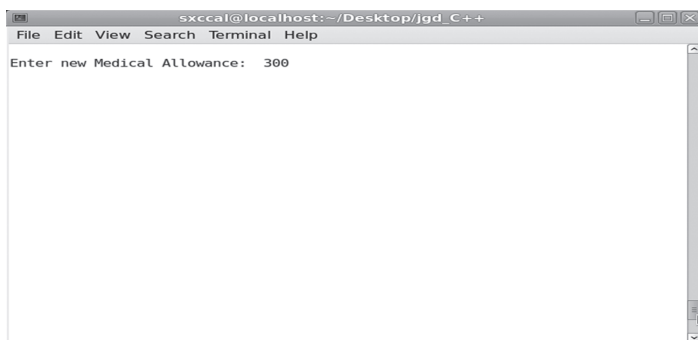
Setting the HRA rate (Option 4 of main menu)



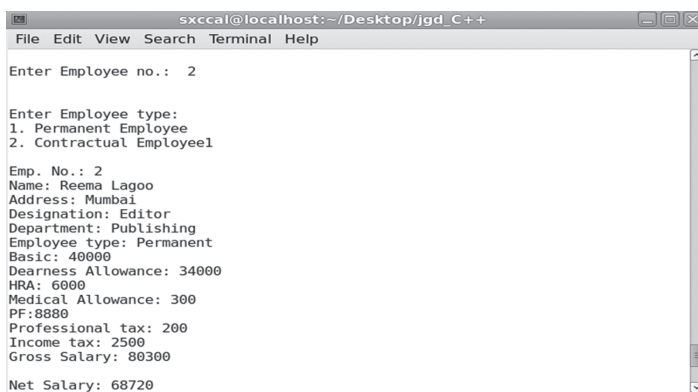
Setting the Professional Tax amount (Option 5 of main menu)



Setting the medical allowance amount (Option 6 of main menu)



Displaying the details of an employee (Option 7 of main menu)



Overtime dues (Option 8 of main menu)

