Acknowledgement

I would like to take this opportunity to thank every individual who has directly or indirectly in some way helped me during the course of this assignment.

Initially I would to give warm thanks to my module lecturer *******, because without whom the aim of successful completion was really impossible. She is very cooperative, supportive by nature. I want to give lot of thanks to her for her suggestion and guidance.

To being with I wish to express my deepest gratitude towards *******, for providing us a platform to achieve my goals and ambitions.

I also like to express our deepest gratitude towards Prof******* for motivating us at every juncture.

Again I would like to give thanks to my friends, seniors for their support in the successful completion of the assignment.

Mohit Kumar

....

Signature

Contents

1 Introduction

This project is based on developing a system which analyze the data of student and print the appropriate grade report for student for APIIT SD INDIA. The objective of the project is to design a window based program which having mainly two users one is administrator and other is student. Administrator has full access to the system and student having limited access i.e. he can only see his/her academic record.

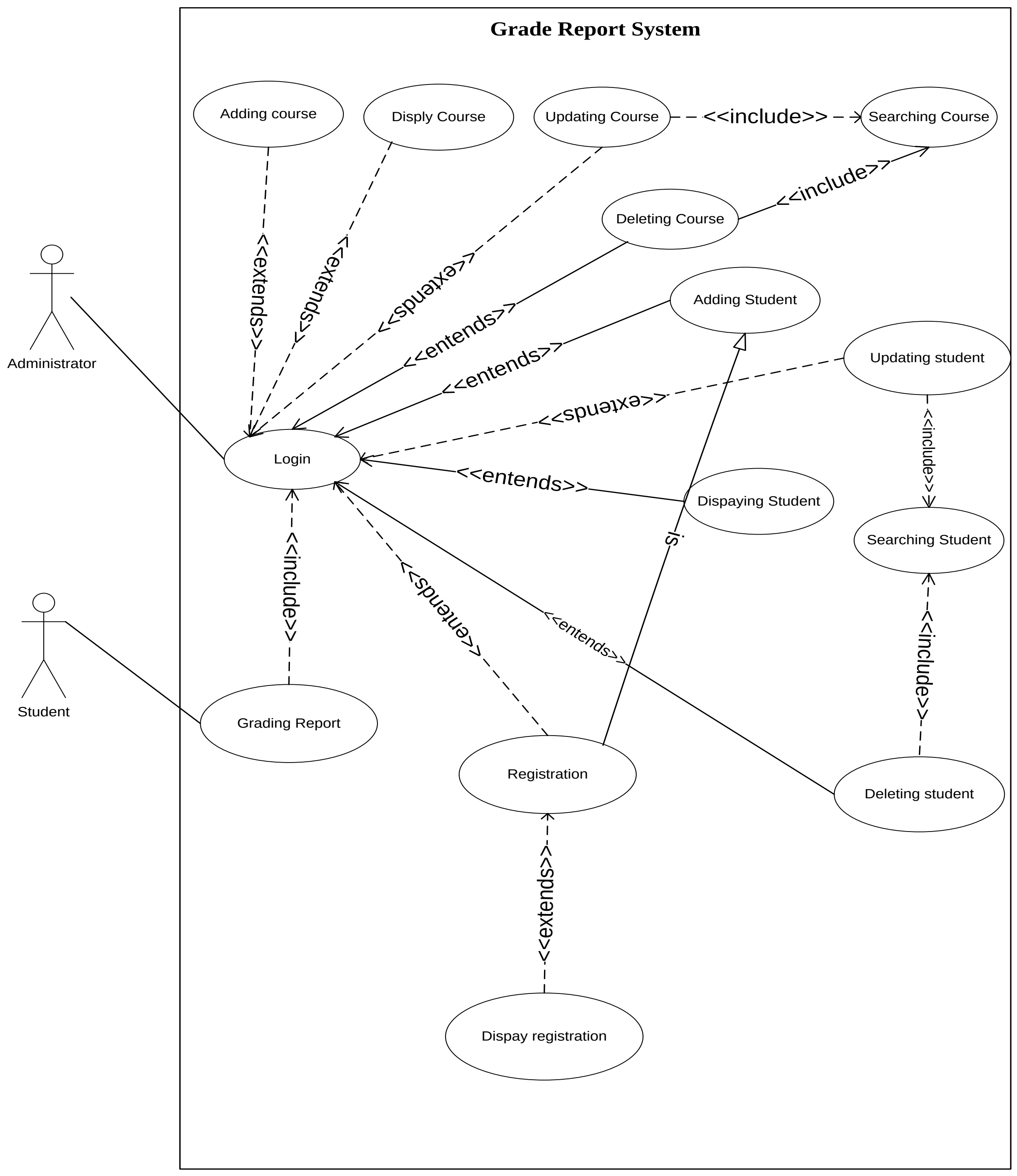
This project is focus on three modules namely login, course and student modules. Through Login module user will identify that which user is interacting with the system. If he is administrator then give full access to user and if user is student then simply display the academic record. Through course and student module administrator can add, display, delete, search and update the records of course and student respectively. The system also calculates tuition fee per credit hour and GPA.

The GPA will display only for those students which are paid their tuition fee. If administrator will login to system then he will see the GPA while student paid the tuition fee or not.

Student module provide many update option to administrator i.e. update student name, update fee paid status, GPA and deletion status. Through update option of course module administrator will add more courses in existing branch and level and change name of existing course also update fee rate.

2 Description and justification of the design of the implementation codes

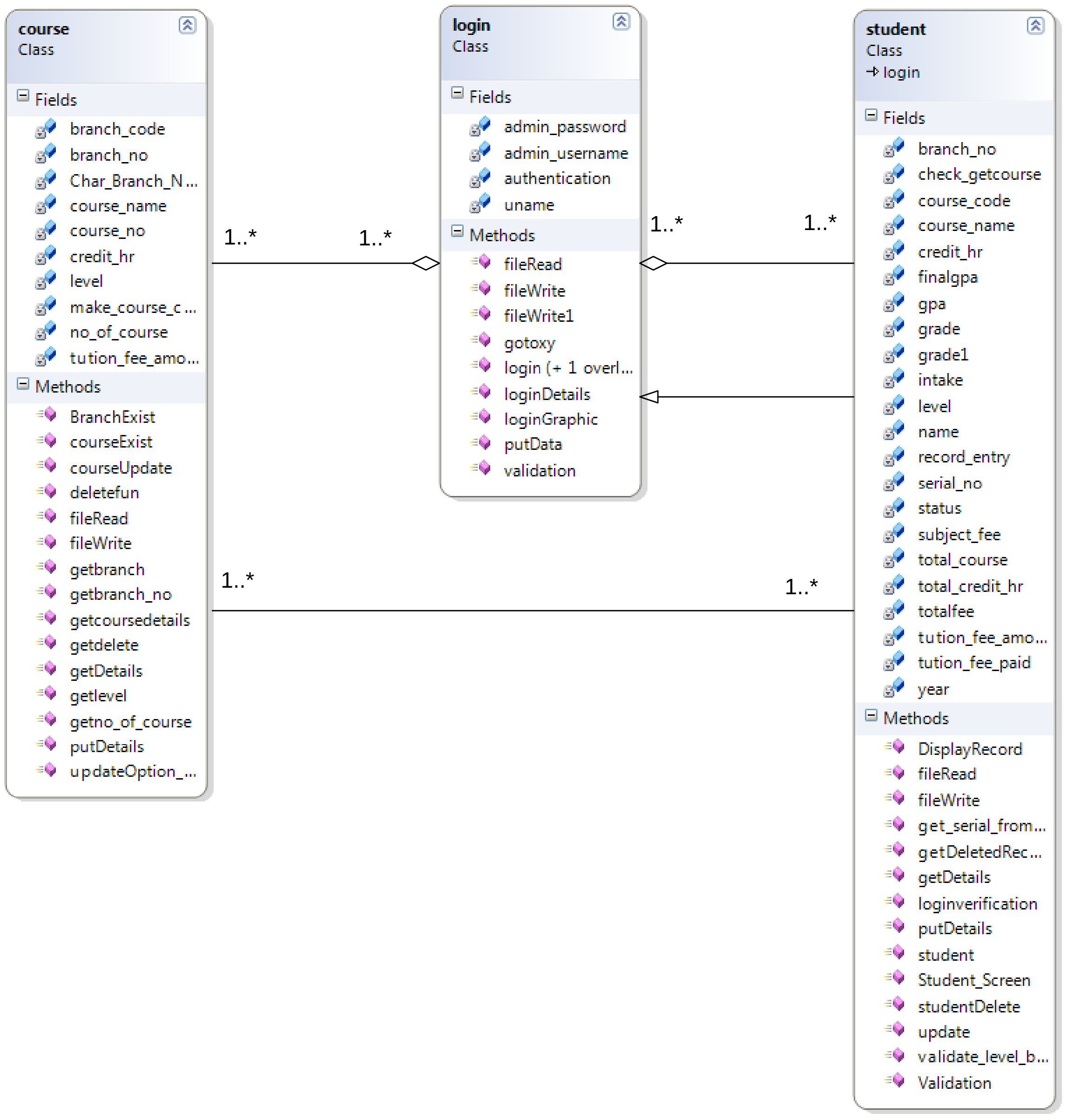
2.1 Usecase diagram



Justification: To use the grading report system administrator and student have to login with the system. There are two different accesses power of user:

1. Administrator: After login administrator have option to perform task related to course management, student management and user management. he can perform add, update display records and delete task of on course and student management. So there are options to work on each module, it show through extend relationship. To update or delete any record of student or course, administrator has to search the records. So there is a include relationship. There is a generalization relationship i.e. "Is a" relationship between student management and user management i.e. student "is a" user of the system. (Ibm.com, 2013)
2. Student: To see the grade report, student have to login with the system. Thus login the per condition for student to see his/her grad report.

2.2 Class Diagram



Justification:

1. Aggregation: Login class contains (call) the student menu with the help of student class object. One object of the login class is associated with one object of student class and many students will login through different object of login class. Login class has same relationship with course class as student class. One student object has one course object and many objects of course is associated with many object of students.
2. Generalisation: Login is base class and student is derived class. With the help of student class object developer call the function of login class. Thus it contains generalisation relationship. (Uml-diagrams.org, 2013)

3 Object Oriented Programming Concept

3.1 Class

Class is collection of similar objects. It is also known as user defined data type. The class contains data member and member function wrapped in it. The variable declare inside the class is called data member and the associated function is known as member function. It is blue print of objects. The class members can be defined with the help of public, private and protected access modifiers. By default members of class are private. (Studytonight.com, 2013)

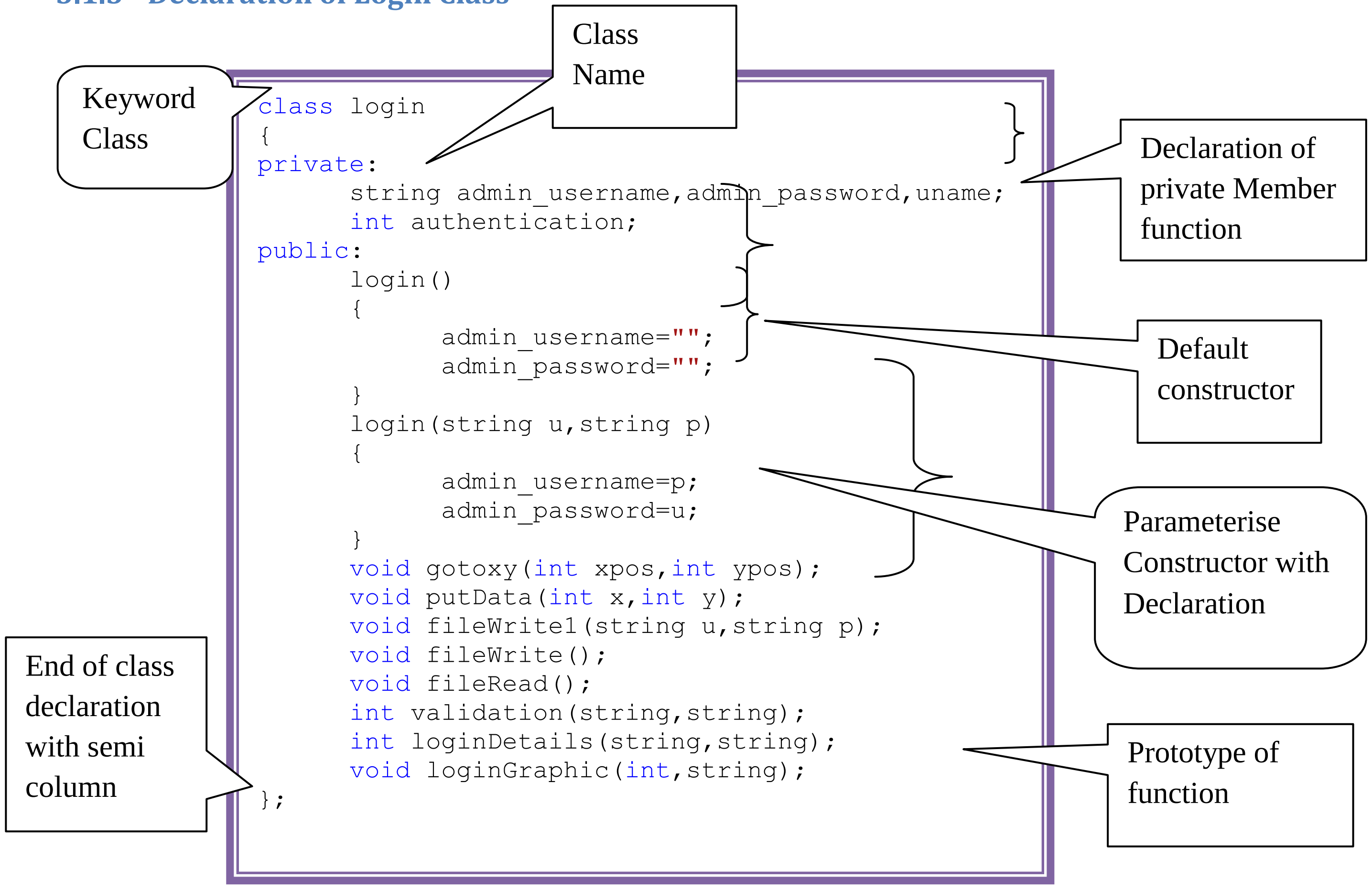
3.1.1 List of classes used in the project

1. Login Class
2. Course Class
3. Student Cass

3.1.2 Description of Login Class

The purpose of this class is to authenticate the user and write the registration details of administrator and student in login file. This class contains member function and data members. Data members are private access specifier and member functions are public. Data members contain declaration of variables and member function having only declaration/prototype of the functions. (Hscripts.com, 2013)

3.1.3 Declaration of Login Class



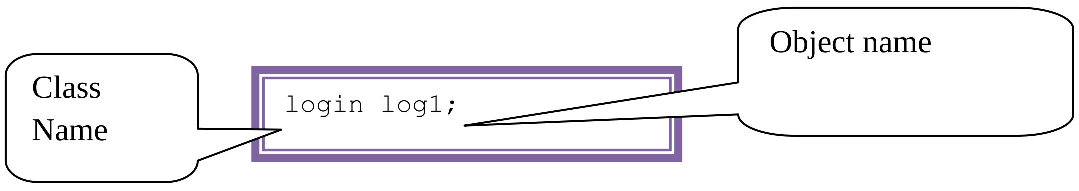
```
class login
{
private:
        string admin_username,admin_password,uname;
        int authentication;
public:
        login()
        {
                admin_username="";
                admin_password="";
        }
        login(string u,string p)
        {
                admin_username=p;
                admin_password=u;
        }
        void gotoxy(int xpos,int ypos);
        void putData(int x,int y);
        void fileWrite1(string u,string p);
        void fileWrite();
        void fileRead();
        int validation(string,string);
        int loginDetails(string,string);
        void loginGraphic(int,string);
};
```

Keyword Class
Class Name
Declaration of private Member function
Default constructor
Parameterise Constructor with Declaration
End of class declaration with semi column
Prototype of function

3.2 Object

Objects are the basic unit of object oriented programming. It is also called instance of the class. The data members and member functions that operate on data are bundled as a unit is called object.

3.2.1 Object of Login Class

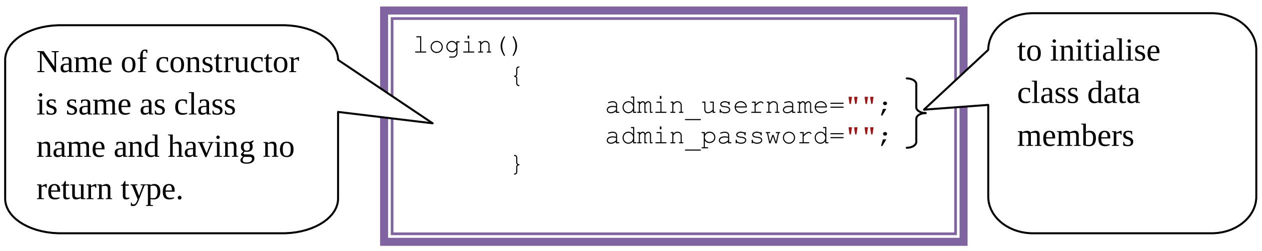


```
login log1;
```

3.3 Default Constructor

Default constructor is a special type of function that is called when object of the class is created. It can be used to initialise class data members. It is implicitly called when object of that class is created. (Deitel and Deitel, 2010, pp. 439-445)

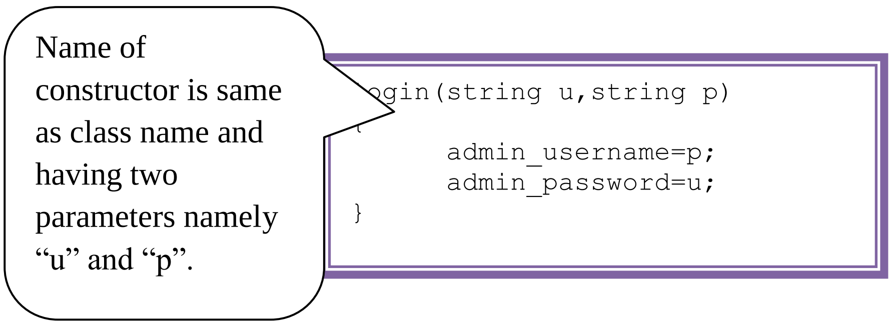
3.3.1 Constructor of Login Class


Name of constructor is same as class name and having no return type.
to initialise class data members

```
login()
      {
                  admin_username="";
                  admin_password="";
      }
```

3.4 Parameterise Constructor:

Constructor having parameter inside it is called parameterise constructor. It is explicitly called.


Name of constructor is same as class name and having two parameters namely “u” and “p”.

```
ogin(string u,string p)
{
        admin_username=p;
        admin_password=u;
}
```

3.5 Overloading:

Function having same name but different number of argument is called function overloading. Overloading is done in the same class. If compiler not found the possible match of signature of function then it go to type conversion of the signature. (Tutorialspoint.com, 2013)

```
class login
{
private:
      string
admin_username,admin_password,uname;
      int authentication;
public:
      login()
      {
            admin_username="";
            admin_password="";
      }
      login(string u,string p)
      {
            admin_username=p;
            admin_password=u;
      }
      void gotoxy(int xpos,int ypos);
      void putData(int x,int y);
      void fileWrite1(string u,string
p);
      void fileWrite();
      void fileRead();
      int validation(string,string);
      int loginDetails(string,string);
      void loginGraphic(int,string);
};
#endif
```

Constructor having same name but different argument is called as constructor overloading.

3.6 Encapsulation

It is binding of data member and member function together in a class. Developer access the private variable of the class through public method of the class.

3.6.1 Encapsulation within Course class

```
class course
{
private:
	char
branch[30],branch_code[10],course_name[20][11],course_
no[10][11];
	int level,no_of_course,branch_no;
	string make_course_code[60],credit_hr[60];
	float tution_fee_amount;
public:
	void getlevel();
	void getbranch();
	void getDetails();
	void getbranch_no();
	void getno_of_course();
	void getcoursedetails();
	void putDetails();
	void updateOption_add(course& c,char ch);
	void updateOption_exist(course& c);
	void fileWrite();
	void fileRead();
	int BranchExist();
	int courseExist(string);
	void getdelete(int n,course& c);
	void courseUpdate();
	friend int getCourse(student& s1, course& c1);
	friend int validate_branch_no(student& s1,
course& c1);
	friend void calculate_GPA(student& s1, course&
c1,int i);
	friend int fee(int,int);
	void deletefun();
};
```

Private data members are accessible through the public method of

3.7 Inheritance

It allows us to reuse the code. The class which is inherited is called base class and inherit class is called as derived class. Derived class inherits features of base class. The derived class is access the non private member of base class. (Tutorialspoint.com, 20[illegible])

Derived class

Base Class

```
class student:public login
```

```
student s1;

s1.fileWrite1(s1.intake,(string)s1.name);
```

With the help of derived class object i.e. student class object we are accessing the base class member function namely “fileWrite” function.

3.8 File Handling

File and stream: The collection of records is called file and stream refer to sequence of bytes. (Bitavoncpp.com, 2013)

3.8.1 Streaming Classes:

ifstream: This class is used to read from file.
Ofstream: This class is used to write in a file.
fstream: This class is used to both read and write purpose. (Balagurusamy, 2011, pp. 290-315)

3.8.2 Creating object of stream class

Name of stream class

```
fstream file;
```

Name of fstream class object

3.8.3 Opening a file with the help of stream class object

```
ifstream fcin;
fcin.open    udent", ios::app |ios::in);
```

Open method is used to open the file with the help of ifstream object. It has two arguments, first is the file name and second is the mode in which we want to open the file.

3.8.4 Write in a file

```
ofstream fout;

fout.wri  ((char *) &c,sizeof(c));
```

With the help of write function of stream class we can write data in a file. Write function write the block of binary data.

3.8.5 Read from the file

```
ifstream fcin;

fcin.read ((char *) &c,sizeof(c));
```

With the help of read method of ifstream class we can read block of binary data from a file.

3.8.6 Close a file

```
fcin.close();
```

3.8.7 To move pointer

1. Seekg function:

```
fcin.seekg(0);
```

Used to move the get pointer of ifstream and having Argument as number of bytes.

2. Seekp function

```
file.seekg((position-sizeof(c)),ios::beg);
```

Used to move the put pointer of ofstream and having Argument as number of bytes and position

3.8.8 Read from file full code

```
void course::fileRead()
{
        course c;
        int i=0;
        ifstream fcin;
        fcin.open("course", ios::app |ios::in);
        fcin.seekg(0) ;
        while(1)
        {
                fcin.read ((char *) &c,sizeof(c));
                if(fcin.eof()!=0)
                break;
                else
                {
                        c.putDetails();
                }
        }
        fcin.close();
        system("pause");
}
```

Creating object of ifstream

Open the course file to read the content from file

Move get pointer to the starting of file

With the help of read function read a full block of data from file.

If pointer reached at end of file then it return non zero value otherwise return zero

Close the current file.

Display read data on user screen.

4 Implementation

4.1 Header File:

User defined Header file consist of two parts. The first part is called include guard. The purpose of include guard is to ignore redeclaration of header file in our program. The second part of the header file contains the actual contents of our program i.e. declaration for other class, function, structure etc.

4.1.1 Login header file

1. **Description:** The header file contains declaration of mainmenu function, Loginmenu function, login class which include declaration of data member and member function and LOGIN_H include guard.
2. **Code snippet :**

```
#ifndef LOGIN_H
#define LOGIN_H
#include <string>
#include<iostream>
void mainmenu();
#include<fstream>
#include<windows.h>
void LoginMenu();
using namespace std;
class login
{
private:
      string admin_username,admin_password,uname;
      int authentication;
public:
      login()
      {
            admin_username="";
            admin_password="";
      }
      login(string u,string p)
      {
            admin_username=p;
            admin_password=u;
      }
      void gotoxy(int xpos,int ypos);
      void putData(int x,int y);
      void fileWrite1(string u,string p);
      void fileWrite();
      void fileRead();
      int validation(string,string);
      int loginDetails(string,string);
      void loginGraphic(int,string);
};
#endif
```

4.1.2 Course Header file:

1. **Description:** This header file includes declaration of coursemenu function and student class ,course class and include guard .
2. **Code snippet :**

```
#ifndef COURSE_H
#define COURSE_H
#include <string>
#include<iostream>
```

```
#include<fstream>
#include "Login.h"
void coursemenu();
class student;
using namespace std;
class course
{
private:
      char
branch[30],branch_code[10],course_name[20][11],course_no[10][11];
      int level,no_of_course,branch_no;
      string make_course_code[60],credit_hr[60];
      float tution_fee_amount;
public:
      void getlevel();
      void getbranch();
      void getDetails();
      void getbranch_no();
      void getno_of_course();
      void getcoursedetails();
      void putDetails();
      void updateOption_add(course& c,char ch);
      void fileWrite();
      void fileRead();
      int BranchExist();
      int courseExist(string);
      void getdelete(int n,course& c);
      void courseUpdate();
      friend int getCourse(student& s1, course& c1);
      friend int validate_branch_no(student& s1, course& c1);
      friend void calculate_GPA(student& s1, course& c1,int i);
      friend int fee(int,int);
      void deletefun();
};
#endif
```

4.1.3 Student Header file:

1. **Description:** The student header file consists of declaration of studentmenu function, course class, student class and also contains STUDENT_H include guard. (Learncpp.com, 2013)
2. **Code snippet:**

```
#ifndef STUDENT_H
#define STUDENT_H
#include<fstream>
#include<string>
#include "course.h"
#include<windows.h>
using namespace std;
void studentmenu();
class course;
class login;
class student:public login
{
private:
      char name[60],grade[5][60];
      string
intake,course_name[60],course_code[60],status,credit_hr[60];
      int
year,level,branch_no,serial_no,total_course,check_getcourse,grade1[60
],gpa,record_entry,total_credit_hr;
```

```
	bool tution_fee_paid;
	float tution_fee_amount,subject_fee[65],finalgpa,totalfee;
public:
	student(){}
	void getDetails(int,int,int,int);
	void putDetails();
	void fileWrite();
	void fileRead(int);
	int Validation(string);
	int validate_level_branch();
	void studentDelete();
	int get_serial_from_file();
	int getDeletedRecord();
	friend void calculate_GPA(student& s1, course& c1,int i);
	friend int validate_branch_no(student& s1, course& c1);
	friend int getCourse(student& s1, course& c1);
	void DisplayRecord(student &s);
	void update();
	void Student_Screen(student &student_detail);
	friend int fee(int,int);
	void loginverification();
};
#endif
```

4.2 Implementation file

4.2.1 Login.cpp

It consists of definition of member function of login class and global declaration of login class object.

1. **putdate Function :**The purpose of this function is to display user name and password on screen. It is member function of login class. It has two argument x and y of data type int and return type of function is void.

1.1. **Code Snippet:**

```
void login::putData(int x,int y)

{

        gotoxy(x+6,y+6);

        cout<<admin_username;

        gotoxy(x+30,y+6);

        cout<<admin_password<<"\n";

}
```

2. **fileWrite function:** Through this function administrator can register the user in the system i.e. write user details in the file. The function return type is of void type, this means the function not return any value and having no argument.

2.1 Code snippet:

```
void login::fileWrite()
{
        int auth=0;
        string un,ppw;
        system("cls");
        ofstream fcout;
        loginGraphic(7,"WELCOME TO GRADING SYSTEM");
        int a=9,b=2;
        gotoxy(a+5,b+2);
        cout<<"Enter User Name: ";
        cin>>un;
        gotoxy(a+22,b+2);
        cout<<"                         ";
        transform(un.begin(), un.end(), un.begin(), toupper);
        gotoxy(a+22,b+2);
        cout<<un;
        gotoxy(a+5,b+4);
        cout<<"Enter Password: ";
        cin>>ppw;
    fcout.open("Login", ios::app |ios::out);
        log=login(ppw,un);
        auth=validation(un,ppw);
        if(auth==1)
        {
                gotoxy(a+5,b+6);
                cout<<"Username Exist please try again\n";
        }
        if(auth==99)
        {
                fcout.write ((char *) &log,sizeof(log));
                gotoxy(a+5,b+6);
                cout<<"write success\n";
        }
        file.close();
        gotoxy(a+5,b+8);
        system("pause");
}
```

3. **fileWrite1 function:** At the time of writting the student record in the file this function is call. Through this functon student user name and password is written in the login file at the time of adding of student record. This function having two argument namely “u” and “p” of string type and return type of this function is void.

3.1. **Code snippet:**

```
void login::fileWrite1(string u,string p)//for student
{
        login log1;
        ofstream fcout;
        transform(u.begin(), u.end(), u.begin(), toupper);
        log1.admin_username=u;
        log1.admin_password=p;
        fcout.open("Login", ios::app |ios::out);
        fcout.write ((char *) &log1,sizeof(log1));
        file.close();

}
```

4. **FileRead Function:** The purpose of this function is to display registration details(User name and password) of student and administrator on the screen.

4.1 Code snippet:

```
void login::fileRead()
{
        file.open("Login", ios::app |ios::in);
        file.seekg(0) ;
        int x=9,y=0;
        for(int i=0;i<27;i++)
        {
                gotoxy(x,y);
                cout<<"*";
                x=x+2;
        }
        x=9;y=4;
        for(int i=0;i<27;i++)
        {
                gotoxy(x,y);
                cout<<"*";
                x=x+2;
        }
        x=9;y=0;
        gotoxy(x+12,y+2);
        cout<<"Welcome To Registration Details";
        x=9;y=2;
        gotoxy(x+6,y+4);
        cout<<"user Name";
        gotoxy(x+30,y+4);
        cout<<"Password";
        while(1)
        {
                file.read ((char *) &log,sizeof(log));
                if(file.eof()!=0)
                {
                        break;
                }
                else
```

```
				{
					log.putData(x,y);
					y=y+2;
				}
			}
			file.close();
			system("pause");
		}
```

5. **Validation function:** The purpose of this function is to validate username name password. This function having two argument namely “un” and “pw” of string type and return type of this function is of integer type which having value 99 and 1. (Cplusplus.com, 2013)

5.1 Code Snippet:

```
int login::validation(string un,string pw)
{
	int i=99;
	login d;
	ifstream fcin;
	fcin.open("Login", ios::ate |ios::in);
	fcin.seekg((sizeof(d)),ios::beg);
	while(1)
	{
		fcin.read ((char *) &d,sizeof(d));
		if(fcin.eof()!=0)
		{
			break;
		}
		if(un==d.admin_username && pw==d.admin_password)
		{
				i=1;
				break;
		}
	}
	fcin.close();
	return i;
}
```

6. **loginDetails function:** Through this function developer check user name and password for administrator. It has two parameters namely “un” and “ppw” of type string and return type of function is int. The function return 1 if user exist in the system and 0 if user does not exist.

6.1 Code snippet:

```
int login::loginDetails(string un,string ppw)
{
	int auth=0;
	fstream file;
	transform(un.begin(), un.end(), un.begin(), toupper);
	file.open("Login", ios::ate |ios::in);
	file.seekg(0,ios::beg);
	file.read ((char *) &log,sizeof(log));
	if(un==log.admin_username && ppw==log.admin_password)
	{
		auth=1;
	}
	file.close();
	return auth;
}
```

4.2.2 Course.cpp

1. **Getlevel function:** This function is used to validate level number entered by user. The return type of the function is void and having no argument.

1.1. **Code Snippet:**

```
void course::getlevel()
{
	int check=0;
	do
	{
		try
		{
			cout<<"\nEnter Level: ";
			cin>>level;
			if(level>=0 && level<4)
				check=1;
		}
		catch(istream::failure e)
		{
			check=0;
			fflush(stdin);
			cin.clear();
		}
	}while(check!=1);
}
```

2. **Getno_of_course Function:** Through this function developer validate total number of course entered by user in one branch.

2.1. **Code Snippet:**

```
void course::getno_of_course()
{
	int check=0,whitespace=0;
	do
		{
			check=1;
			try
			{
				cout<<"\nPlease Enter New Tution fee rate: RS
";

				cin>>tution_fee_amount;
				cout<<"number of courses you want to enter:
";

				cin>>no_of_course;
				if(no_of_course>0 && no_of_course<=10)
					check=1;
			}
			catch(istream::failure e)
			{
				check=0;
				fflush(stdin);
				cin.clear();
			}
		}while(check!=1);
}
```

3. **getDetails function:** This function work same as getDetail function of login class. The return type of this function is void and having no argument.

3.1. **Code Snippet:**

```
void course::getDetails()
```

```
{
        Similar to previous  “getDetail” function of login class
}
```

4. **PutDetails function:** This function used to display information about course on the screen. The return type of function is void and having no argument.

4.1.**Code Snippet:**

```
 void course::putDetails()
 {
      Work same as putdata of login class
 }
```

5. **fileWrite function:** The function is used to write the course details in the course file. The return type of function is void and having no argument.

5.1.**Code Snippet:**

```
 void course::fileWrite()
 {
      work same as fileWrite function of class login
 }
```

6. **fileRead function:** The function is used to read the course details from the course file. The return type of function is void and having no argument.

6.1.**Code Snippet:**

```
 void course::fileRead()
 {
      work same as fileRead function of class login
 }
```

7. **BranchExist function:** Through this function developer check the level and branch exixtance. If the level and branch exixt hen it return 1 and if not exist it return 0. The return type of function is void and having no argument.

7.1.**Code Snippet:**

```
int course::BranchExist()
{
     Work same as Validation function of class login
}
```

8. **CourseExist Function:** This function is used to check the couse name , if course name exist in the level then will return the index value and if not exist return 0 value to the function. The return type of function is int and having one arguments namely “search_course_nm” of string type.

8.1.**Code snippet:**

```
int course::BranchExist()
{
     Work same as Validation function of class login
}
```

9. **courseUpdate function:** This function used to change/update the details of course through menu provided by developer. The return type of function is of void type and having no argument.

9.1.**Code Snippet:**

```
void course::courseUpdate()
{
      system("cls");
      int choice;
```

```
	int j,i;
	char char_i[10];
	char 
char_level[10][11],char_branch_no[10][11],char_last[10][11];
	string course_nm[30],course_Num[10],search_course_nm;
	course c;
	fstream file;
	int flag=0,position=0;
	login lod;
	lod.loginGraphic(6,"WELCOME TO UPDATE MENU");
	int a=6,y=4;
	lod.gotoxy(a+6,y);
	cout<<"1. Add New course\t2.Change in Existing one";
	lod.gotoxy(a+6,y+2);
	cout<<"3. update Tution fee Rate\t4.Exit";
	lod.gotoxy(a+8,y+4);
	cout<<"Enter your option: ";
	cin>>choice;
	if(choice!=4)
	{
		system("cls");
		lod.loginGraphic(8,"WELCOME TO UPDATE MENU");
		lod.gotoxy(a+6,y);
		cout<<"Enter Level: ";
		cin>>level;
		lod.gotoxy(a+6,y+2);
		cout<<"Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE";
		lod.gotoxy(a+15,y+4);
		cout<<"4.Mechatronic  5.AUTOMOBILE    6.EE";
		lod.gotoxy(a+6,y+8);
		cout<<"enter Branch Number: ";
		cin>>branch_no;
	}
	flag=BranchExist();
	file.open("course",ios::in|ios::out|ios::ate);
	file.seekg(0);
	while(1 && flag==1)
	{
		file.read ((char *) &c,sizeof(c));
		if(file.eof()!=0)
		{
			break;
		}
		if(level==c.level && branch_no==c.branch_no)
		{
			position=file.tellg();
			if(choice==2)
			{
				cout<<"\nEnter course name which you want to 
update: ";
				cin>>search_course_nm;
				i=0;
				j=c.courseExist(search_course_nm);
				if(j!=0)
				{
					i=j-1;
					make_course_code[i]="";
					cout<<"\n\t"<<i+1<<". enter course 
name: ";
					cin>>c.course_name[i];
					itoa(level,char_level[i],10);
```

```
                              itoa(branch_no,char_branch_no[i],10);
                              itoa(j,char_i,10);
                              strcpy(course_no[i],"");
                              strcat(course_no[i],char_level[i]);
                              strcat(course_no[i],char_branch_no[i]);
                              strcat(course_no[i],char_i);
                              course_nm[i]=(string)c.course_name[i];
                              transform(course_nm[i].begin(),
course_nm[i].end(),course_nm[i].begin(), toupper);
                              course_nm[i]=course_nm[i].substr(0,3);

      make_course_code[i].append(course_nm[i]);

      make_course_code[i].append(course_no[i]);

      c.make_course_code[i]=make_course_code[i];
                              cout<<"\n\tCourse code:
"<<c.make_course_code[i];
                              cout<<"\n\n\t"<<i+1<<". Credit Hour for
"<<c.course_name[i]<<" :";
                              cin>>c.credit_hr[i];
                              file.seekg((position-
sizeof(c)),ios::beg);
                              file.write((char*)&c,sizeof(c));
                              cout<<"\n\t@@@@@@@@@@@@@@--Course
Updated Successfully--@@@@@@@@@@@@@@@\n";
                              system("pause");
                              courseUpdate();
                        }
                        else
                        {
                              cout<<"\n@@@@@@@@@@@@@@@@@@Course Not
exist@@@@@@@@@@@@@\n";
                              courseUpdate();
                        }
                  }
                  if(choice==3)
                  {
                        lod.loginGraphic(11,"WELCOME TO UPDATE
MENU");
                        lod.gotoxy(a+6,y+10);
                        cout<<"Current tution fee rate:
"<<c.tution_fee_amount;
                        lod.gotoxy(a+6,y+12);
                        cout<<"Please Enter New rate: ";
                        cin>>c.tution_fee_amount;
                        file.seekg((position-sizeof(c)),ios::beg);
                        file.write((char*)&c,sizeof(c));
                        lod.gotoxy(a+6,y+16);
                        cout<<"Rate Updated Successfully";
                        lod.gotoxy(a+6,y+18);
                        system("pause");
                        courseUpdate();
                  }
                  if(choice!=4)
                  {
                        system("cls");

      cout<<"\n*******************************************************
********\n";
```

```
                        cout<<"\##              Add New course In 
level: "<<c.level<<"           ##\n";

      cout<<"\n*****************************************************
********\n";
                        c.putDetails();
                  }
                break;
            }
      }
      if(choice==1 && flag==1)
      {
            if(c.no_of_course<=10)
            {
                  c.updateOption_add(c,'y');
                  file.seekg((position-sizeof(c)),ios::beg);
                  file.write((char*)&c,sizeof(c));
                  cout<<"@@@@@@@@@@@@@@@@@--Course Added 
Successfully--@@@@@@@@@@@@@@@@@@@@@\n";
                  system("pause");
                  courseUpdate();
            }
            else
            {
                  cout<<"\n@@@@@@@@@@@@@@@--Maximum Course Limit is 
Riched--@@@@@@@@@@@@@@@@@\n";
            }
      }

      if(choice>=5 && flag==1)
      {
            cout<<"\n\t@@@@@@@@@@@@--Invalid selection--
@@@@@@@@@@@@@@@\n";
            system("pause");
            courseUpdate();
      }
      if(choice==4)
            coursemenu();
      if(flag==0)
      {
            cout<<"\nBranch not Exist\n";
      }
      file.close();
}
```

2. **Deletefun function:** Function used to delete specific course from the course file. The return type of the function isof void type and having no argument.

2.1.**Code Snippet:**

```
void course::deletefun()
{
  int flag2=0,j=0;
  int flag,m,flag12,position;
  string file_course_nm2,line,msg="DELETE COURSE";
  course c;
  login lod;
  lod.loginGraphic(7,msg);
  int a=5,y=4;
  fstream file;
  string search_course_nm;
  lod.gotoxy(a+8,y);
  cout<<"Eneter Level: ";
```

```
cin>>level;
lod.gotoxy(a+8,y+2);
cout<<"Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE";
lod.gotoxy(a+14,y+4);
cout<<"4.Mechatronic  5.MBA    6.EE";
lod.gotoxy(a+8,y+6);
cout<<"enter Branch Number: ";
cin>>branch_no;
lod.gotoxy(a+8,y+8);
cout<<"enter course name: ";
cin>>search_course_nm;
flag=courseExist(search_course_nm);
if(flag==0)
{
      lod.gotoxy(a+8,y+10);
      cout<<"course not exist";
      lod.gotoxy(a+8,y+12);
      system("pause");
      coursemenu();
}
if(flag!=0)
{
      file.open("course",ios::in|ios::out|ios::ate);
      file.seekg(0);
      while(1)
      {
            file.read ((char *) &c,sizeof(c));
            if(file.eof()!=0)
            {
                  break;
            }
            if(level==c.level && branch_no==c.branch_no)
            {
                  flag12=1;
                  position=file.tellg();
                  for(int i=0;i<c.no_of_course;i++)
                  {
                        if(i==flag-1)
                        {
                              flag2=1;
                              i=i-1;
                              flag=99;
                        }
                        else
                        {

                              if(flag2==0)
                              {
                                    j=i;

strcpy(c.course_name[i],c.course_name[j]);

strcpy(c.course_no[i],c.course_no[j]);

c.make_course_code[i]=c.make_course_code[j];

c.credit_hr[i]=c.credit_hr[j];

                              }
                              else
                              {
                                      j=i+1;

  strcpy(c.course_name[i],c.course_name[j]);

  strcpy(c.course_no[i],c.course_no[j]);

  c.make_course_code[i]=c.make_course_code[j];

  c.credit_hr[i]=c.credit_hr[j];
                                }
                          }
                          j++;
                    }
                    c.no_of_course=c.no_of_course-1;;
                    lod.gotoxy(a+8,y+10);
                    cout<<"@@@@@@@@@Record deleted@@@@@@@@";
                    file.seekg((position-sizeof(c)),ios::beg);
                    file.write((char*)&c,sizeof(c));
                    file.close();
                    break;
              }
        }
  }
  else
  {

  }
}
```

4.2.3 Student.cpp

1. **Student_Screen function:** This function is called when student will login to the system and display his academic details. The function contains one argument of student class data type and return type of function is void.

1.1.**Code Snippet:**

```
void student::Student_Screen(student &student_detail)
{
	system("cls");
	string notpaid="***";
	int x=1,y=2,a=9,b=0;
	for(int i=0;i<27;i++)
	{
		gotoxy(a,b);
		cout<<"*";
		a=a+2;
	}
	a=9;b=0;
	gotoxy(a+20,1);
	cout<<"Welcome "<<name;
	for(int i=0;i<27;i++)
	{
		gotoxy(a,b+2);
		cout<<"*";
		a=a+2;
	}
	a=9;b=0;
	for(int i=0;i<10;i++)
	{
		gotoxy(a,b+2);
		cout<<"*";
		b=b+2;
	}
	for(int i=0;i<27;i++)
	{
		gotoxy(a,b+2);
		cout<<"*";
		a=a+2;
	}
	a=61;b=0;
	for(int i=0;i<10;i++)
	{
		gotoxy(a,b+2);
		cout<<"*";
		b=b+2;
	}


	x=1;y=1;
	gotoxy(x+12,y+2);
	fflush(stdin);
	cout<<"Student Name:";
	gotoxy(x+45,y+2);
	fflush(stdin);
	puts(student_detail.name);
	gotoxy(x+12,y+4);
	cout<<"Level:";
	gotoxy(x+45,y+4);
	cout<<student_detail.level;
```

```
gotoxy(x+12,y+6);
cout<<"Student ID:";
gotoxy(x+45,y+6);
cout<<student_detail.intake;
gotoxy(x+12,y+8);
cout<<"Number of Courses enrolled:";
gotoxy(x+45,y+8);
cout<<student_detail.check_getcourse;
gotoxy(x+12,y+10);
cout<<"Course No";
gotoxy(x+27,y+10);
cout<<"Course Name";
gotoxy(x+42,y+10);
cout<<"Credits";
gotoxy(x+52,y+10);
cout<<"Grade";
int y1=14;
if(student_detail.tution_fee_paid==1)
{
      for(int i=0;i<student_detail.check_getcourse;i++)
      {
            if(student_detail.subject_fee[i]==0)
            {

            }
            else
            {
                  gotoxy(x+12,y1);
                  cout<<student_detail.course_code[i];
                  gotoxy(x+27,y1);
                  cout<<student_detail.course_name[i];
                  gotoxy(x+42,y1);
                  cout<<student_detail.credit_hr[i];
                  gotoxy(x+52,y1);
                  cout<<student_detail.grade[i];
                  y1=y1+2;
            }
      }
      gotoxy(x+12,y1);
      cout<<"Total Number of Credit:";
      gotoxy(x+42,y1);
      cout<<student_detail.total_credit_hr;
      gotoxy(x+27,y1+2);
      cout<<"GPA:";
      gotoxy(x+43,y1+2);
      cout<<student_detail.finalgpa;
}
else
{
      for(int i=0;i<check_getcourse;i++)
      {
            if(s1.subject_fee[i]==0)
            {

            }
            else
            {
                  gotoxy(x+12,y1);
                  cout<<student_detail.course_code[i];
                  gotoxy(x+27,y1);
                  cout<<student_detail.course_name[i];
```

```
                        gotoxy(x+42,y1);
                        cout<<student_detail.credit_hr[i];
                        gotoxy(x+52,y1);
                        cout<<notpaid;
                        y1=y1+2;
                }
            }
            gotoxy(x+12,y1);
            cout<<"Total Number of Credit:";
            gotoxy(x+42,y1);
            cout<<student_detail.total_credit_hr;
            gotoxy(x+27,y1+2);
            cout<<"GPA:";
            gotoxy(x+43,y1+2);
            cout<<notpaid;
            gotoxy(x+12,y1+4);
            cout<<"Pay your free, to see your grade";
            gotoxy(x+12,y1+6);
            cout<<"Due Fee: RS "<<student_detail.totalfee;
        }
        cout<<"\n\n\n";
}
```

2. **Fee Function:** This function is used to access the private member of course classs. Through this function developer brings the tuition fee from the course file and save it to student file. It has two argument namely “level” and “branch” of int type and the return type of function is int.

2.1.**Code Snippet:**

```
int fee(int level,int branch_no)
{
        int i=0;
        ifstream fcin;
        fcin.open("course", ios::app |ios::in);
        fcin.seekg(0);
        while(1)
        {
                fcin.read ((char *) &c1,sizeof(c1));
                if(fcin.eof()!=0)
                {
                        break;
                }
                if(level==c1.level && branch_no==c1.branch_no)
                {
                        i=c1.tution_fee_amount;
                    break;
                }
                else
                {
                        i=0;
                }
        }
        fcin.close();
        return i;
}
```

3. **getDeleteRecord Function:** The developer use this function to validate the limit of student in a branch. The function having no argument and return type of function is int.

3.1.**Code Snippet:**

```
int student::getDeletedRecord()
{
        int i=0;
        student sd;
        ifstream fcin;
        fcin.open("student", ios::app |ios::in);
        fcin.seekg(0);
        while(1)
        {
                fcin.read ((char *) &sd,sizeof(sd));
                if(fcin.eof()!=0)
                {
                        break;
                }
                if(sd.status.compare("deactive")==0)
                {
                        i++;
                }
        }
        fcin.close();
        return i;
}
```

4. **Get_serial_from_file Function:** Through this function developer get the serial number of the student added at the last execution of the system in the same level and branch and get the maximum value of the serial number from the student file. The function having no argument and having return type is int.

4.1.**Code Snippet:**

```
int student::get_serial_from_file()
{
        int i=0;
        int j=0,max=0;
        int serial[70];
        student sd;
        ifstream fcin;
        fcin.open("student", ios::app |ios::in);
        fcin.seekg(0);
        while(1)
        {
                fcin.read ((char *) &sd,sizeof(sd));
                if(fcin.eof()!=0)
                {
                        break;
                }
                if(level==sd.level && branch_no==sd.branch_no &&
year==sd.year)
                {
                        serial[j]=sd.serial_no;
                        if(max<serial[j])
                        {
                                max=serial[j];
                        }
                        j++;
                }
        }
        fcin.close();
        return max+1;
}
```

5. **getCourse Function:** It is friend function of student class and course class. Developer uses this function to get course details from the course file and save details in student file. Also he calculates the GPA of student. The Function contains argument as object of student class and course class and return type of function is int.

5.1.**Code Snippet:**

```
int getCourse(student& s1, course& c1)
{
	ifstream fcin;
	s1.total_credit_hr=0;
	int i,flag1=0,checkgrade=0;
	string s_level,c_level,s_branch,c_branch,s_coursename[100];
	char 
ch_s_level[5],ch_s_branch[5],ch_c_branch[5],course_level[10];
	fcin.open("course",ios::app|ios::in);
	fcin.seekg(0);
	while(1)
	{
		fcin.read((char*)&c1,sizeof(c1));
		if(fcin.eof()!=0)
			break;
		else
		{
			s1.gpa=0;
			s_level=string(itoa(s1.level,ch_s_level,10));
			c_level=string(itoa(c1.level,course_level,10));
			s_branch=string(itoa(s1.branch_no,ch_s_branch,10));
			c_branch=string(itoa(c1.branch_no,ch_c_branch,10));
			if(s_level==c_level && s_branch==c_branch)
			{
				s1.tution_fee_amount=c1.tution_fee_amount;
				s1.total_course=c1.no_of_course;
				flag1=c1.no_of_course;
				for(i=0;i<c1.no_of_course;i++)
				{

	s1.course_name[i]=(string)c1.course_name[i];

	s1.course_code[i]=(string)c1.make_course_code[i];
					calculate_GPA(s1,c1,i);
				}
				break;
			}
		}
	}
	fcin.close();
	return flag1;
}
```

6. **Calculate_GPA Function:** Developer used this function to calculate total fee for each subject, total credit hour and sum of grade points for a single student. He is then calculating final GPA in “getDetail” function. The Return type of the function is void and the argument as object of student class and course class.

6.1.**Code Snippet**

```
void calculate_GPA(student& s1, course& c1,int i)
{
	int checkgrade=0;
     s1.credit_hr[i]=c1.credit_hr[i];
	char *credit_hour=new char[(s1.credit_hr[i]).size()+1];
	credit_hour[s1.credit_hr[i].size()]=0;
	memcpy(credit_hour,(s1.credit_hr[i]).c_str(),(s1.credit_hr[i]).
size());
	s1.subject_fee[i]=atoi(credit_hour)*s1.tution_fee_amount;
	s1.totalfee=s1.totalfee+s1.subject_fee[i];
```

```
        s1.total_credit_hr=s1.total_credit_hr+atoi(credit_hour);
        do
        {
                        cout<<"\n\tEnter Grade course
"<<s1.course_name[i]<<" (A-F): ";
                        cin>>s1.grade[i];

                if(string(s1.grade[i]).compare("A")==0
||string(s1.grade[i]).compare("a")==0
||string(s1.grade[i]).compare("B")==0
||string(s1.grade[i]).compare("b")==0
||string(s1.grade[i]).compare("C")==0
||string(s1.grade[i]).compare("c")==0
||string(s1.grade[i]).compare("D")==0
||string(s1.grade[i]).compare("d")==0
||string(s1.grade[i]).compare("E")==0
||string(s1.grade[i]).compare("e")==0
||string(s1.grade[i]).compare("F")==0
||string(s1.grade[i]).compare("f")==0);
                {
                        if(string(s1.grade[i]).compare("A")==0
||string(s1.grade[i]).compare("a")==0)
                        {
                                s1.grade1[i]=5;
                        }
                         if(string(s1.grade[i]).compare("B")==0
||string(s1.grade[i]).compare("b")==0)
                                s1.grade1[i]=4;
                         if(string(s1.grade[i]).compare("C")==0
||string(s1.grade[i]).compare("c")==0)
                                s1.grade1[i]=3;
                         if(string(s1.grade[i]).compare("D")==0
||string(s1.grade[i]).compare("d")==0)
                                s1.grade1[i]=2;
                         if(string(s1.grade[i]).compare("E")==0
||string(s1.grade[i]).compare("e")==0)
                                s1.grade1[i]=1;
                         if(string(s1.grade[i]).compare("F")==0
||string(s1.grade[i]).compare("f")==0)
                                s1.grade1[i]=0;
                        s1.gpa=(atoi(credit_hour)*s1.grade1[i]+s1.gpa);
                        checkgrade=1;
                }

        }while(checkgrade!=1);
}
```

7. **Loginverification Function:** Developer use this function to authenticate the user through there user name and password. This function call the base class i.e. login class member function namely "loginDetails". It has no return type and also contains no argument as parameter. (Asciitable.com, 2013)

7.1.**Code Snippet:**

```
void student::loginverification()
{
        int f=0,check=0,counter=0;
        string un,ppw;
        char p,ppw1[40];
        fflush(stdin);
        student student_detail;
        loginGraphic(7,"WELCOME TO GRADING SYSTEM");
        int a=9,b=2;
        gotoxy(a+5,b+2);
        cout<<"Enter User Name: ";
        cin>>un;
        gotoxy(a+22,b+2);
        cout<<"                        ";
        transform(un.begin(), un.end(), un.begin(), toupper);
        gotoxy(a+22,b+2);
        cout<<un;
        gotoxy(a+5,b+4);
        cout<<"Enter Password: ";
        while(1)
        {

              p=getch();
              if(p>=48 && p<=57)
              {
                    strcpy(ppw1,"");
                    un="";
                    gotoxy(a+5,b+6);
                    cout<<"Wrong input! Please enter only alphabet(a-
  z)\n";
                    gotoxy(a+5,b+8);
                    system("pause");
                    system("cls");
                    loginverification();
                    break;

              }
              else
              {

                    if(p==13)
                          break;
                    else
                    {
                          if(p==8 && counter>0)
                          {
                                printf("\b ");
                                counter--;
                                printf("\b");
                          }
                          else
                          {
                                cout<<"*";
                                ppw1[counter]=p;
                                counter++;
                          }
                    }
              }
        }
        ppw1[counter]='\0';
        ppw=(string)ppw1;
        login log=login(un,ppw);
        ifstream fcin;
        f=student_detail.loginDetails(un,ppw);
        if(f==1)
        {
              mainmenu();
        }
        if(f==0)
        {
                fcin.open("student", ios::app |ios::in);
                fcin.seekg(0);
                while(1)
                {
                        fcin.read ((char *)
&student_detail,sizeof(student_detail));
                        if(fcin.eof()!=0)
                        {
                                break;
                        }
                        if(un.compare(student_detail.intake)==0 &&
ppw.compare((string)student_detail.name)==0)
                        {
                                check=1;
                                system("cls");

        student_detail.Student_Screen(student_detail);
                                system("pause");
                                break;
                        }
                }
                fcin.close();
                if(check==0)
                {
                        gotoxy(a+5,b+6);
                        cout<<"User name or password wrong !Please try
again\n";
                }
                else
                {
                        gotoxy(a+5,b+6);
                        cout<<"User name or password wrong !Please try
again\n";
                }
        }

}
```

4.3 Driver File

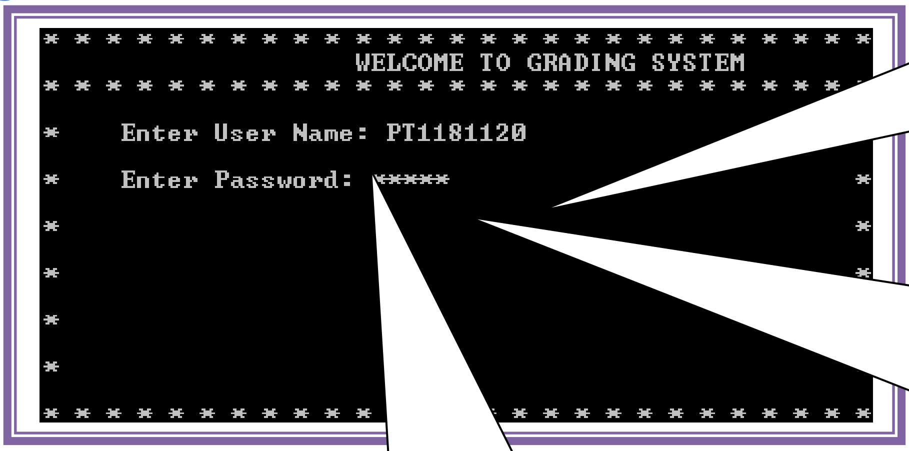
Driver contains main function of the program.

Code Snippet:

```
#include "Login.h"
#include "course.h"
#include "student.h"
int main()
{
 student s;
 s.loginverification();
 return 0;
}
```

5 Sample Output

5.1 Login Screen



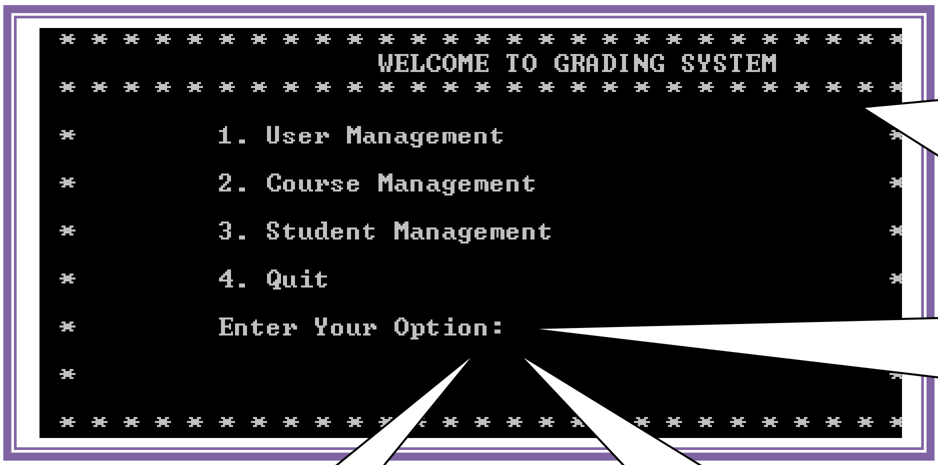
```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                    WELCOME TO GRADING SYSTEM
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*    Enter User Name: PT1181120
*    Enter Password: *****                           *
*                                                    *
*                                                    *
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Step 1: It is first screen of the system. User will enter his user name and password.

Step 2: Entered data will check and if match then user will successful login to system and if entered wrong input then system will close.

Users have to enter alphabetical letter in password

5.2 Administrator Main Menu



```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                    WELCOME TO GRADING SYSTEM
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*          1. User Management                        *
*          2. Course Management                      *
*          3. Student Management                     *
*          4. Quit                                   *
*          Enter Your Option:
*                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

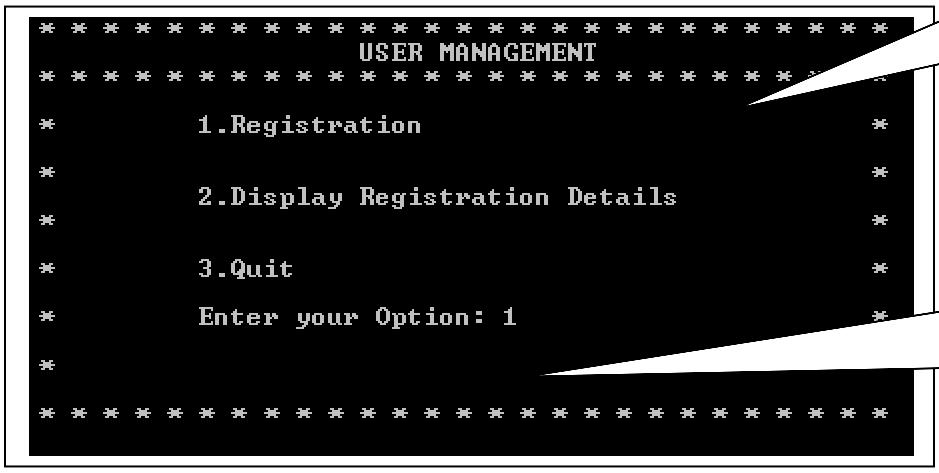
If username and password matched with admin user name and password then this screen will display to administrator.

User can enter integer value and respective menu will be open.

If user input 4 then system will close.

If user entered wrong input then this menu reappear on the screen.

5.3 User Management menu:

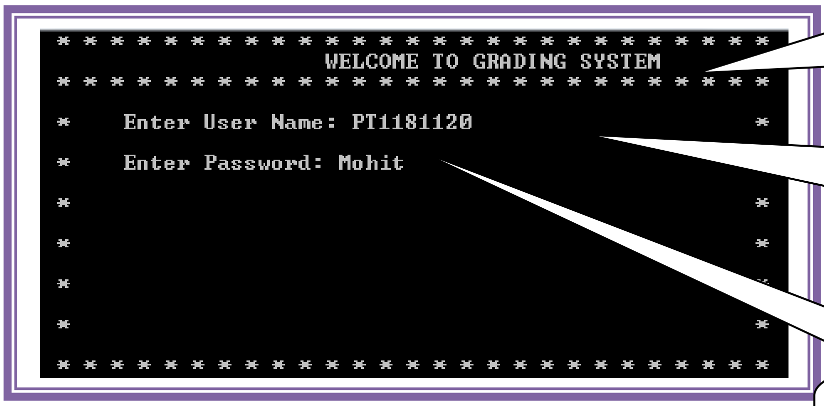


```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                  USER MANAGEMENT
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*          1.Registration                           *
*          2.Display Registration Details           *
*          3.Quit                                   *
*          Enter your Option: 1                     *
*
* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

If admin select 1 from main menu then this menu will open.

User can enter integer value and respective menu will be open.

1. Registration menu:



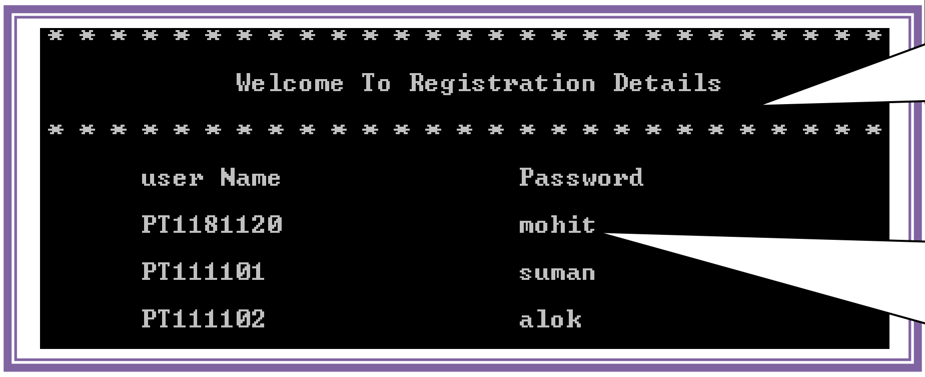
```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
              WELCOME TO GRADING SYSTEM
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*    Enter User Name: PT1181120                     *
*    Enter Password: Mohit
*                                                   *
*                                                   *
*
*                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

If admin will input 1 in user management menu then this menu will open.

With the help of this menu admin will register user to system.

Administrator will enter user name and password for registration of user.

2. Display Registration Details

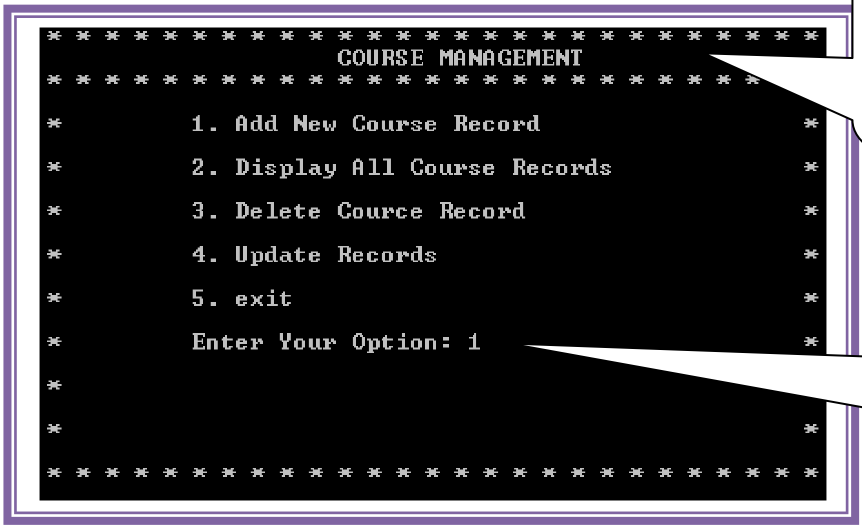


```
* * * * * * * * * * * * * * * * * * * * * * * * * * *

          Welcome To Registration Details

* * * * * * * * * * * * * * * * * * * * * * * * * * *

     user Name                Password
     PT1181120                mohit
     PT111101                 suman
     PT111102                 alok
```

If admin will input 2 in user management menu then this menu will open.

First line of this screen show the detail for administrator and rest of line are registration detail for user.

5.4 Course management

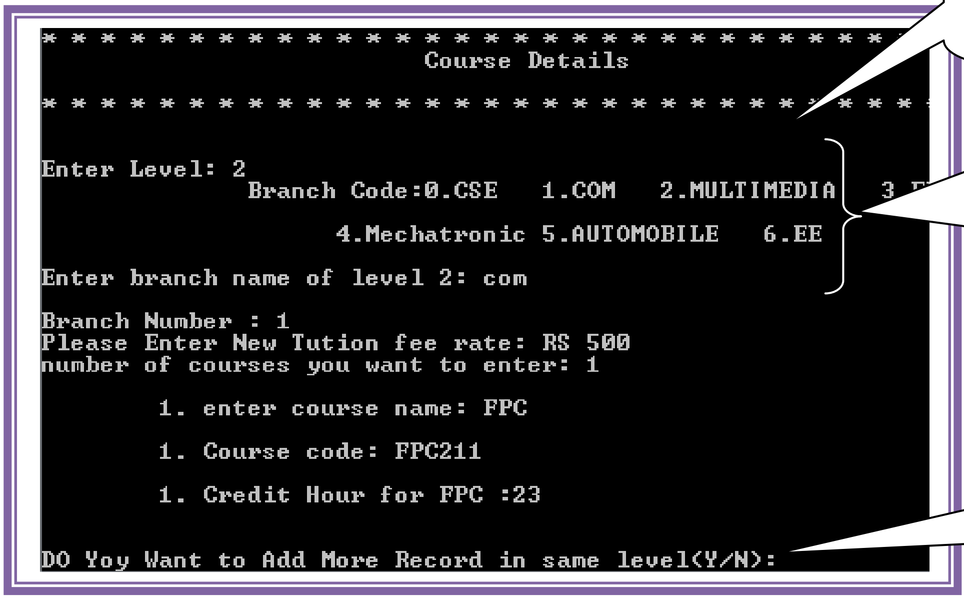


```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                    COURSE MANAGEMENT
* * * * * * * * * * * * * * * * * * * * * * * * * * *

*          1. Add New Course Record                  *

*          2. Display All Course Records             *

*          3. Delete Cource Record                   *

*          4. Update Records                         *

*          5. exit                                   *

*          Enter Your Option: 1                      *

*

*                                                    *

* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

This is course menu for administrator. With the help of this screen admin can perform his action according to menu option

Use can input integer value from 1 to 5 and respective action will perform.

1. Add New Course Record



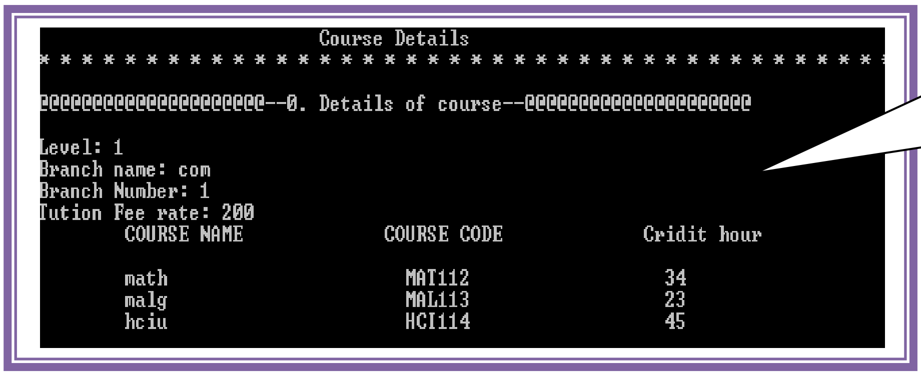
```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                         Course Details

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


Enter Level: 2
              Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3

                    4.Mechatronic 5.AUTOMOBILE   6.EE

Enter branch name of level 2: com

Branch Number : 1
Please Enter New Tution fee rate: RS 500
number of courses you want to enter: 1

        1. enter course name: FPC

        1. Course code: FPC211

        1. Credit Hour for FPC :23


DO Yoy Want to Add More Record in same level(Y/N):
```

Through this screen admin can add course details in course file.

Admin will enter level and course name and respective course number automatically taken and display on screen

If admin want to add new details then he should enter y or n.

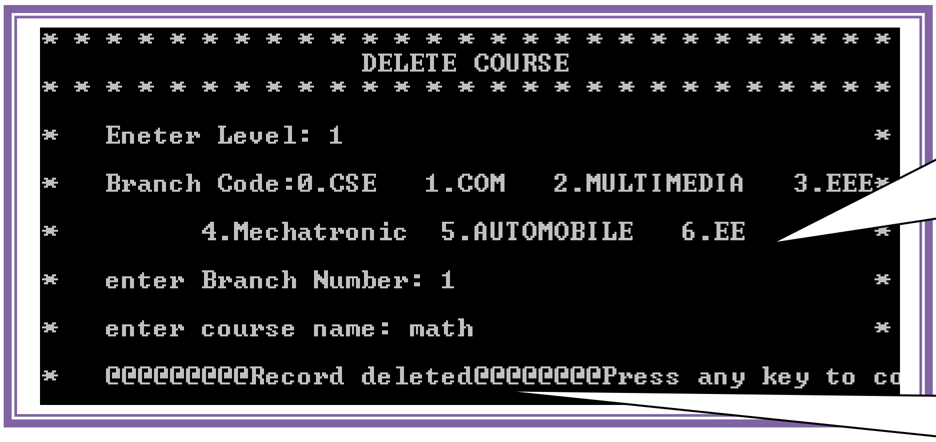
2. Display Course Records



```
                           Course Details
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

@@@@@@@@@@@@@@@@@@@@@--0. Details of course--@@@@@@@@@@@@@@@@@@@@@

Level: 1
Branch name: com
Branch Number: 1
Tution Fee rate: 200
        COURSE NAME             COURSE CODE             Cridit hour

        math                      MAT112                  34
        malg                      MAL113                  23
        hciu                      HCI114                  45
```

This menu show all the records of course file.

3. Delete Course

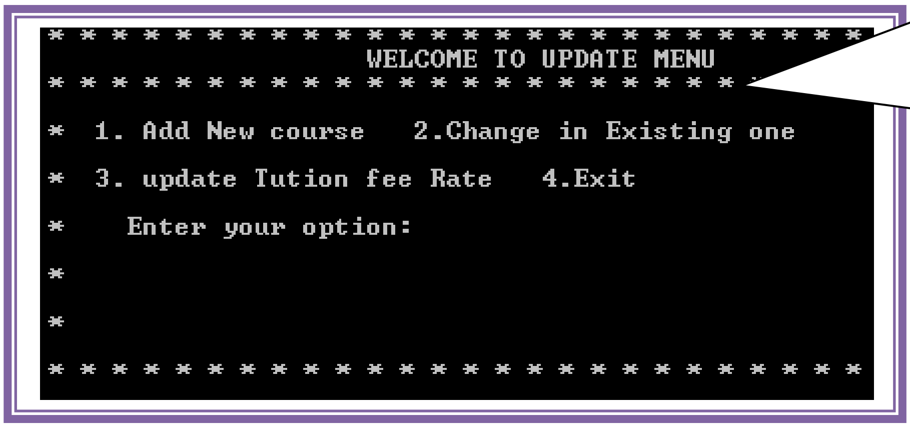


```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                    DELETE COURSE
* * * * * * * * * * * * * * * * * * * * * * * * * * *

*   Eneter Level: 1                                  *

*   Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE*

*         4.Mechatronic  5.AUTOMOBILE   6.EE         *

*   enter Branch Number: 1                           *

*   enter course name: math                          *

*   @@@@@@@@@Record deleted@@@@@@@@Press any key to co
```

Through this screen admin can delete a single course from the entered level and branch.

Deletion success or failure message will display on screen.

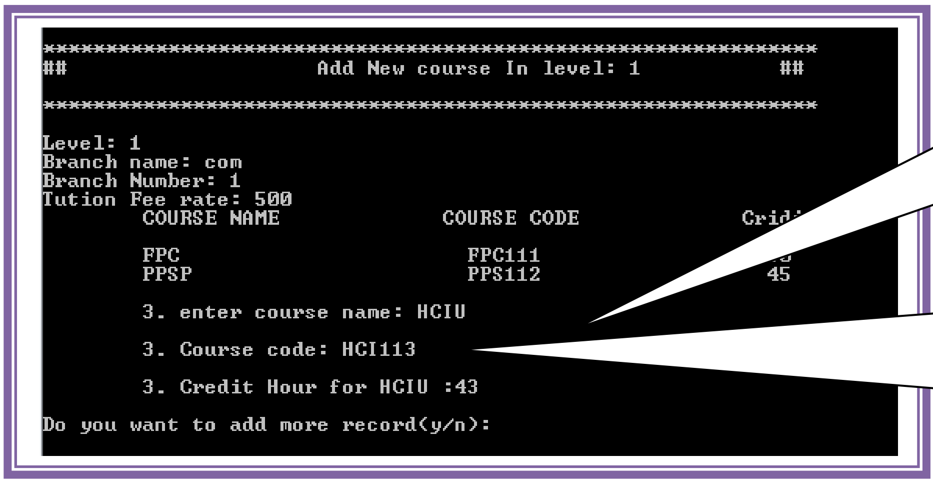
4. Update Record



```
* * * * * * * * * * * * * * * * * * * * * * * * * *
                    WELCOME TO UPDATE MENU
* * * * * * * * * * * * * * * * * * * * * * * *

*  1. Add New course   2.Change in Existing one

*  3. update Tution fee Rate   4.Exit

*    Enter your option:

*

*

* * * * * * * * * * * * * * * * * * * * * * * * * *
```

If user input 4 in course management menu this menu will open. Here user can update the records of the course according to option provided.

4.1.Add new course screen

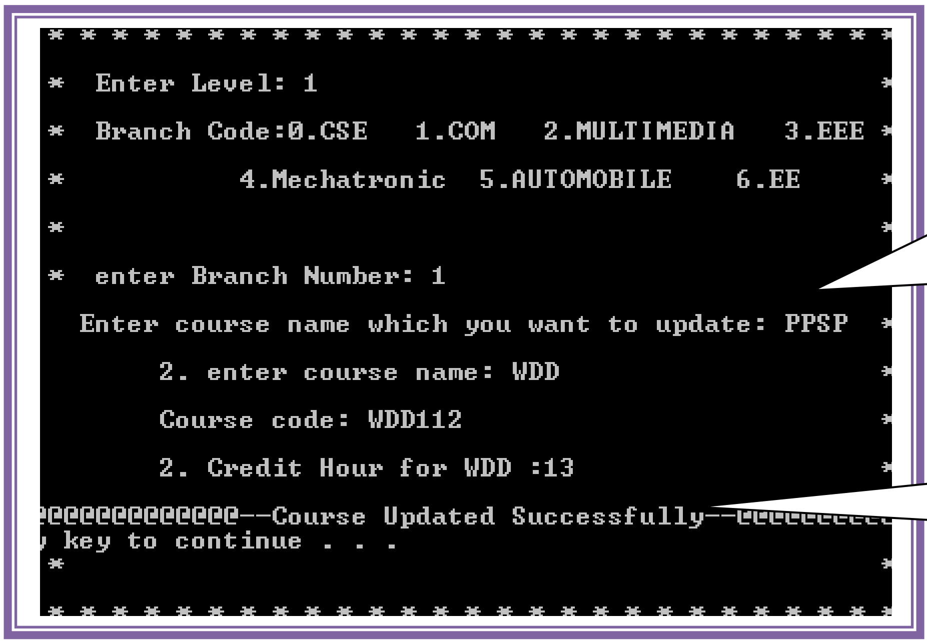


```
***************************************************************
##                    Add New course In level: 1           ##

***************************************************************

Level: 1
Branch name: com
Branch Number: 1
Tution Fee rate: 500
        COURSE NAME              COURSE CODE              Crid

        FPC                        FPC111
        PPSP                       PPS112                   45

        3. enter course name: HCIU

        3. Course code: HCI113

        3. Credit Hour for HCIU :43

Do you want to add more record(y/n):
```

Admin can add new course in existing branch and level.

If admin wants to add more courses in same level and branch then he can input y otherwise input n.

4.2.Change in existing course

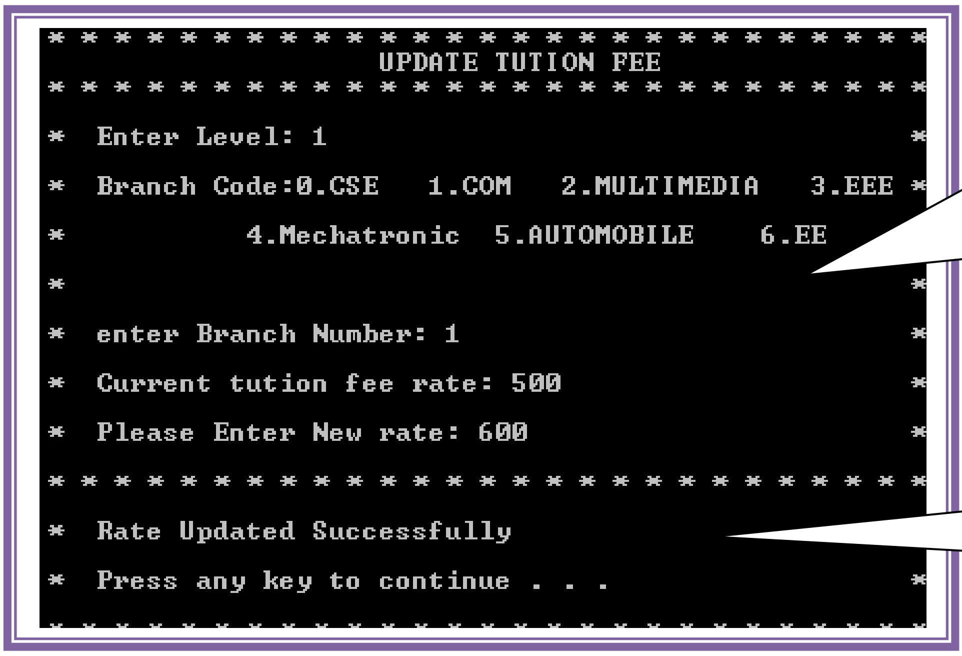


```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Enter Level: 1                                    *
*  Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE *
*           4.Mechatronic  5.AUTOMOBILE    6.EE      *
*                                                    *
*  enter Branch Number: 1
  Enter course name which you want to update: PPSP  *
        2. enter course name: WDD                    *
        Course code: WDD112                          *
        2. Credit Hour for WDD :13                   *
@@@@@@@@@@@@@--Course Updated Successfully--@@@@@@@@@
y key to continue . . .
*                                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Through this menu admin will change/update the existing course name.

Success or failure message will display

4.3.Update tuition fee rate

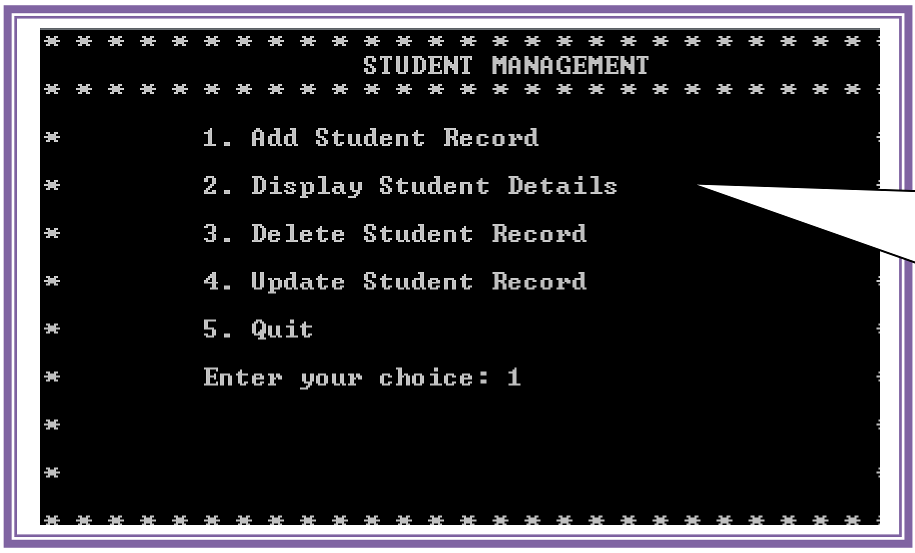


```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                 UPDATE TUTION FEE
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Enter Level: 1                                     *
*  Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE *
*           4.Mechatronic  5.AUTOMOBILE    6.EE
*                                                     *
*  enter Branch Number: 1                             *
*  Current tution fee rate: 500                       *
*  Please Enter New rate: 600                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Rate Updated Successfully
*  Press any key to continue . . .                    *
```

Admin input the level and branch number then tuition fee will display on screen and ask for new tuition fee.

Success or failure message will display

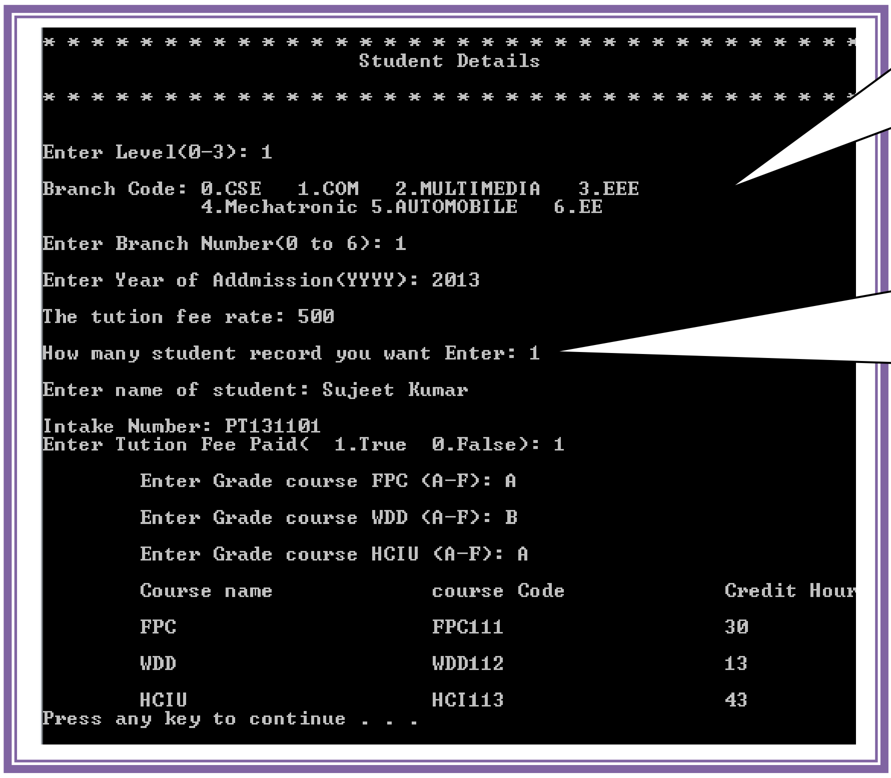
5.5 Student Management menu



```
* * * * * * * * * * * * * * * * * * * * * * * * * *
                 STUDENT MANAGEMENT
* * * * * * * * * * * * * * * * * * * * * * * * * *
*         1. Add Student Record
*         2. Display Student Details
*         3. Delete Student Record
*         4. Update Student Record
*         5. Quit
*         Enter your choice: 1
*
*
* * * * * * * * * * * * * * * * * * * * * * * * * *
```

This is Student menu for administrator. With the help of this screen admin can perform his action according to menu option

1. **Add student record**

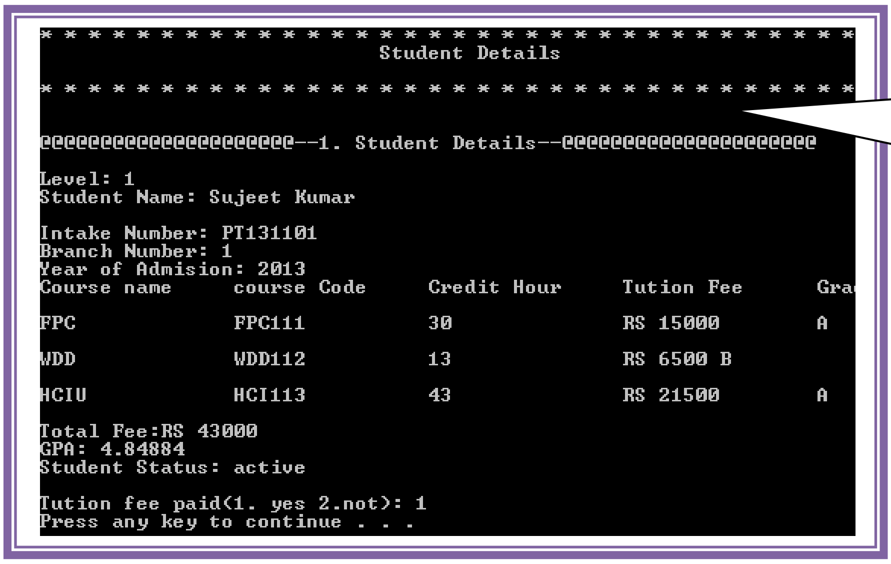


```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                          Student Details

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


Enter Level(0-3): 1

Branch Code: 0.CSE   1.COM   2.MULTIMEDIA   3.EEE
             4.Mechatronic 5.AUTOMOBILE   6.EE

Enter Branch Number(0 to 6): 1

Enter Year of Addmission(YYYY): 2013

The tution fee rate: 500

How many student record you want Enter: 1

Enter name of student: Sujeet Kumar

Intake Number: PT131101
Enter Tution Fee Paid(  1.True  0.False): 1

        Enter Grade course FPC (A-F): A

        Enter Grade course WDD (A-F): B

        Enter Grade course HCIU (A-F): A

        Course name             course Code             Credit Hour

        FPC                     FPC111                  30

        WDD                     WDD112                  13

        HCIU                    HCI113                  43
Press any key to continue . . .
```

Through this screen admin will add the student record in the same sequence.

Here admin enter number of student entry he wants to add in the same level and branch.

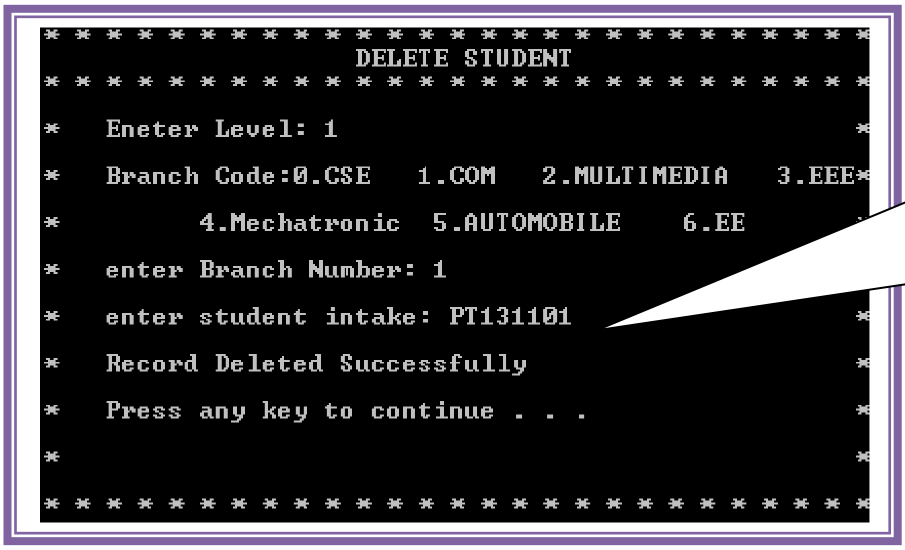
2. **Display student Details**



```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                            Student Details

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


@@@@@@@@@@@@@@@@@@@@@--1. Student Details--@@@@@@@@@@@@@@@@@@@@@@

Level: 1
Student Name: Sujeet Kumar

Intake Number: PT131101
Branch Number: 1
Year of Admision: 2013
Course name      course Code      Credit Hour      Tution Fee      Gra

FPC              FPC111           30               RS 15000        A

WDD              WDD112           13               RS 6500 B

HCIU             HCI113           43               RS 21500        A

Total Fee:RS 43000
GPA: 4.84884
Student Status: active

Tution fee paid(1. yes 2.not): 1
Press any key to continue . . .
```

This screen will display all the student record to admin.

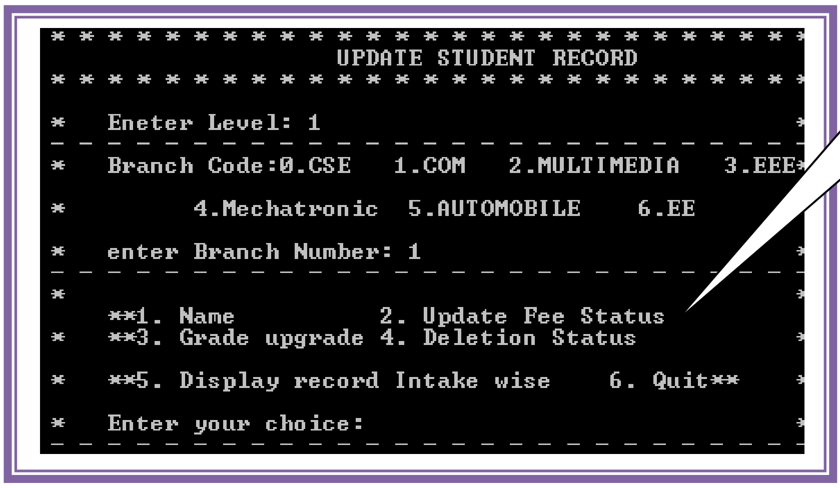
3. **Delete Student record**



```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                    DELETE STUDENT
* * * * * * * * * * * * * * * * * * * * * * * * * * *

*   Eneter Level: 1                                   *

*   Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE*

*         4.Mechatronic  5.AUTOMOBILE    6.EE

*   enter Branch Number: 1

*   enter student intake: PT131101                    *

*   Record Deleted Successfully                       *

*   Press any key to continue . . .                   *

*                                                     *

* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Admin will enter the level and branch number and intake of the student then student status will be change from active to deactivate

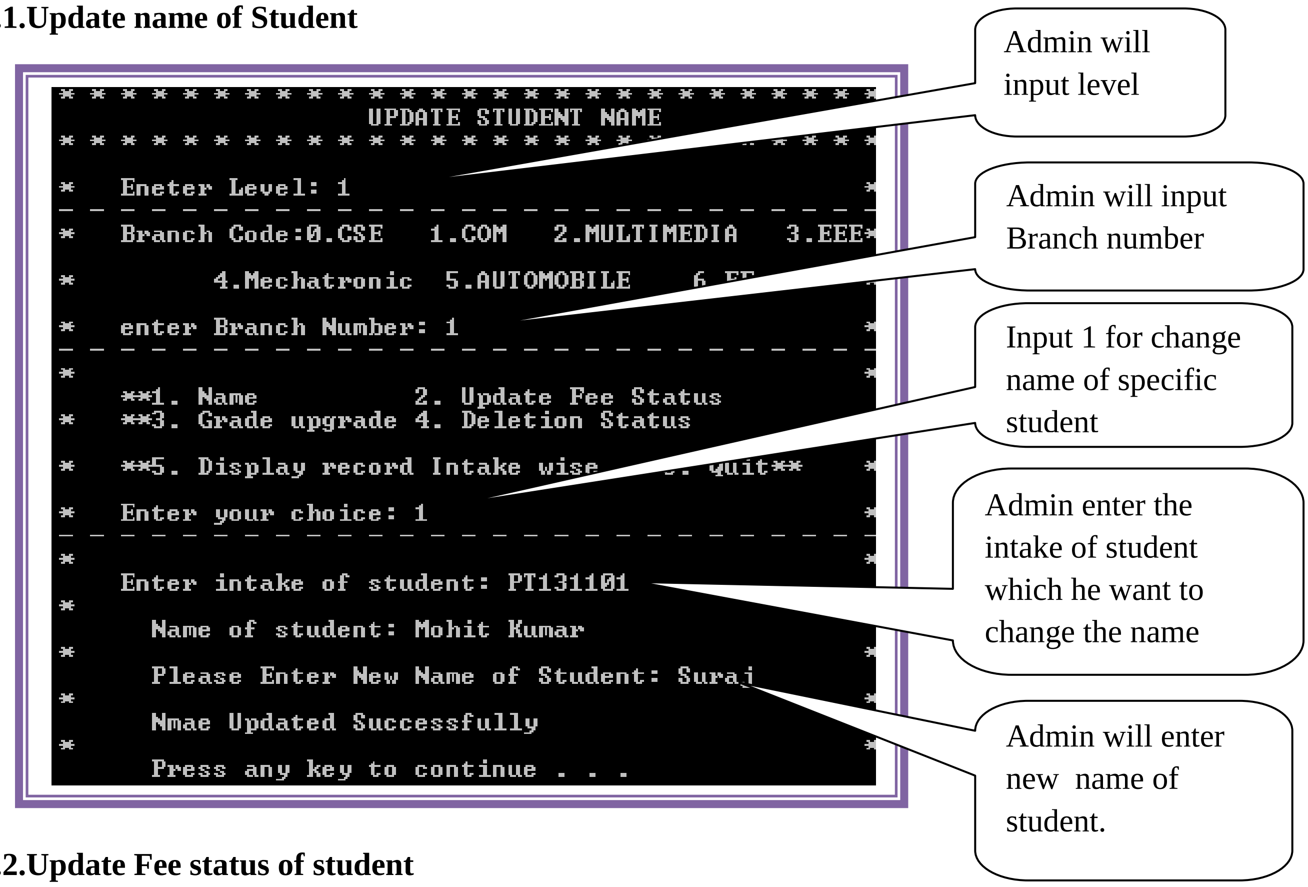
4. **Update menu of student**



```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                  UPDATE STUDENT RECORD
* * * * * * * * * * * * * * * * * * * * * * * * * * *

*   Eneter Level: 1                                   *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*   Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE*

*         4.Mechatronic  5.AUTOMOBILE    6.EE

*   enter Branch Number: 1                            *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*                                                     *
    **1. Name           2. Update Fee Status
*   **3. Grade upgrade  4. Deletion Status            *

*   **5. Display record Intake wise    6. Quit**      *

*   Enter your choice:                                *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Through this menu admin will change /update records of student

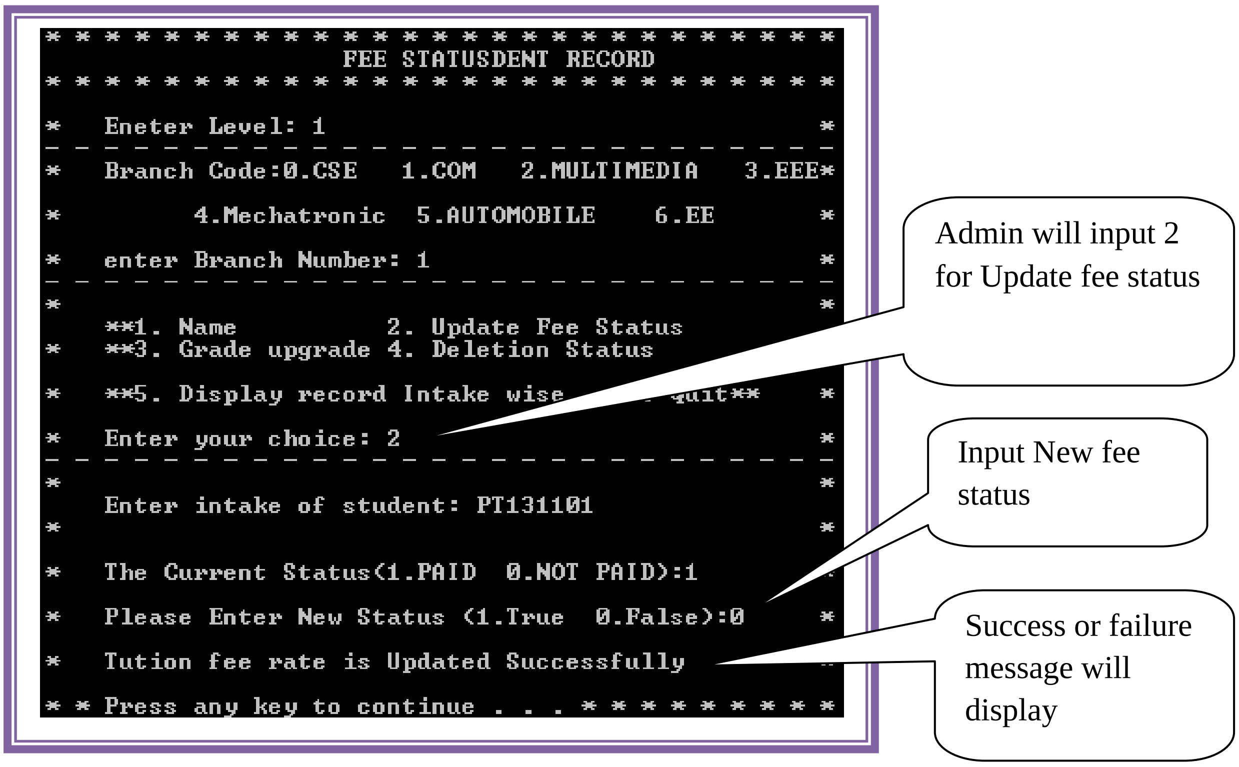
4.1.Update name of Student



```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                    UPDATE STUDENT NAME
* * * * * * * * * * * * * * * * * * * * * * * * * * *

*   Eneter Level: 1                                    *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*   Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE*

*         4.Mechatronic  5.AUTOMOBILE    6.EE          *

*   enter Branch Number: 1                             *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*                                                      *
    **1. Name          2. Update Fee Status
*   **3. Grade upgrade 4. Deletion Status

*   **5. Display record Intake wise     6. quit**      *

*   Enter your choice: 1                               *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*                                                      *
    Enter intake of student: PT131101
*
      Name of student: Mohit Kumar
*                                                      *
      Please Enter New Name of Student: Suraj
*                                                      *
      Nmae Updated Successfully
*                                                      *
      Press any key to continue . . .
```

Admin will input level
Admin will input Branch number
Input 1 for change name of specific student
Admin enter the intake of student which he want to change the name
Admin will enter new name of student.

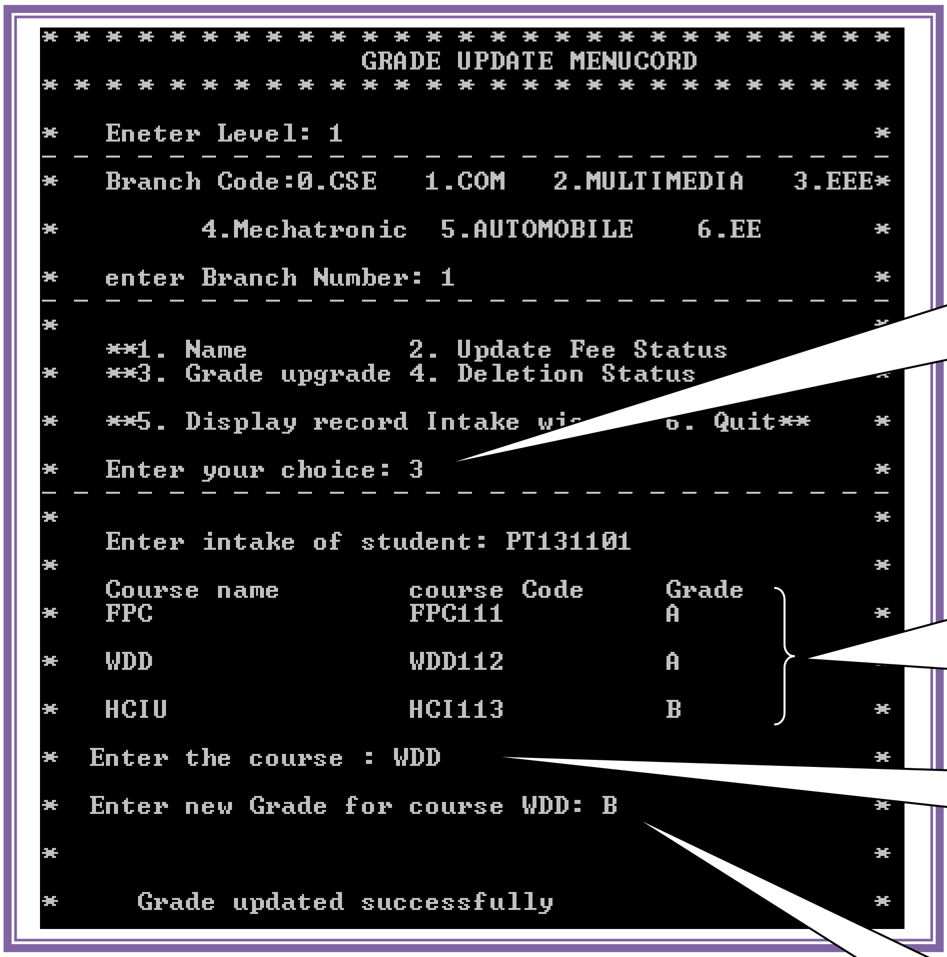
4.2.Update Fee status of student



```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                  FEE STATUSDENT RECORD
* * * * * * * * * * * * * * * * * * * * * * * * * * *

*   Eneter Level: 1                                    *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*   Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE*

*         4.Mechatronic  5.AUTOMOBILE    6.EE          *

*   enter Branch Number: 1                             *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*                                                      *
    **1. Name          2. Update Fee Status
*   **3. Grade upgrade 4. Deletion Status

*   **5. Display record Intake wise     6. quit**      *

*   Enter your choice: 2                               *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*                                                      *
    Enter intake of student: PT131101
*                                                      *

*   The Current Status(1.PAID  0.NOT PAID):1           *

*   Please Enter New Status (1.True  0.False):0        *

*   Tution fee rate is Updated Successfully            *

* * Press any key to continue . . . * * * * * * * * * *
```

Admin will input 2 for Update fee status
Input New fee status
Success or failure message will display

4.3.Update Grade of Student



```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                    GRADE UPDATE MENUCORD
* * * * * * * * * * * * * * * * * * * * * * * * * * *

*   Eneter Level: 1                                 *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*   Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE*

*         4.Mechatronic  5.AUTOMOBILE    6.EE       *

*   enter Branch Number: 1                          *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*
    **1. Name           2. Update Fee Status
*   **3. Grade upgrade 4. Deletion Status

*   **5. Display record Intake wis       6. Quit**  *

*   Enter your choice: 3                            *
- - - - - - - - - - - - - - - - - - - - - - - - - - -
*                                                   *
    Enter intake of student: PT131101
*                                                   *
    Course name         course Code     Grade
*   FPC                 FPC111          A           *

*   WDD                 WDD112          A

*   HCIU                HCI113          B           *

*  Enter the course : WDD                           *

*  Enter new Grade for course WDD: B                *

*                                                   *

*     Grade updated successfully                    *
```

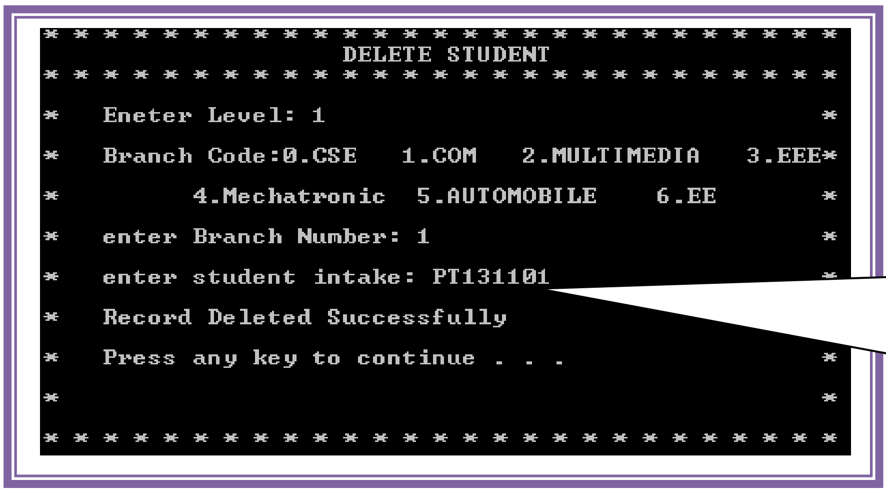
For update the grade of the student admin have to input 3.

Current grade will display automatically on screen after input Intake of student

Enter the course name to update grade

Admin will input new grade of course entered by him.

4.4.Update Deletion status

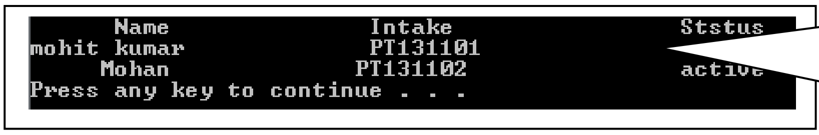


```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
                     DELETE STUDENT
* * * * * * * * * * * * * * * * * * * * * * * * * * *

*   Eneter Level: 1                                 *

*   Branch Code:0.CSE   1.COM   2.MULTIMEDIA   3.EEE*

*         4.Mechatronic  5.AUTOMOBILE    6.EE       *

*   enter Branch Number: 1                          *

*   enter student intake: PT131101                  *

*   Record Deleted Successfully

*   Press any key to continue . . .                 *

*                                                   *

* * * * * * * * * * * * * * * * * * * * * * * * * * *
```

From here admin will enter level, branch number and intake of the student to update stats of student.

Through this screen admin will see the name, intake and status of the

4.5.**Display record of student intake wise**



```
      Name                Intake                    Ststus
mohit kumar             PT131101
     Mohan             PT131102                    active
Press any key to continue . . .
```

6 Conclusion

Developer has successfully made the project. The project contains add, search, update, deletion functionality of the student and course. Through this project developer learn more about window based programming. With the help of this project developer applied OOPs concept of cpp i.e. class, object, encapsulation, inheritance, friend function concept in real word senior. The project helps the developer to understand the file handling concept in cpp. During the project developer were busy to study about the four OOPs concept of cpp and other basic concepts namely encapsulation, inheritance and data handling. The project helped the developer to know about the conversion of one data type to other.

During this project developer face many problems and to solve those problems he has to read different book related to CPP programming. From this he gain the knowledge and it help him to clarify his logics and concepts of CPP.

This project helps developer to understand usecase diagram, class diagram and their respective relationship.

7 References

Agilemodeling.com. 2013. *Introduction to UML 2 Class Diagrams*. [online] Available at: http://www.agilemodeling.com/artifacts/classDiagram.htm [Accessed: 23 Oct 2013].

Asciitable.com. 2013. *Ascii Table - ASCII character codes and html, octal, hex and decimal chart conversion*. [online] Available at: http://www.asciitable.com/ [Accessed: 18 Oct 2013].

Balagurusamy, E. 2011. *Object Oriented Programming With c++*. 5th ed. New Delhi: Tata McGraw Hill Education Private Limited.

Bitavoncpp.com. 2013. *file handling / data handling in c++*. [online] Available at: http://www.bitavoncpp.com/file-handling.html [Accessed: 16 Oct 2013].

Cplusplus.com. 2013. *User input verification. - C++ Forum*. [online] Available at: http://www.cplusplus.com/forum/beginner/28223/ [Accessed: 14 Oct 2013].

Deitel, P. and Deitel, H. 2010. *C++ How to program*. 7th ed. New Delhi: PHI Learning private limited.

Hscripts.com. 2013. *Classes - C++ OOPs concept Tutorial*. [online] Available at: http://hscripts.com/tutorials/cpp/cpp-classes.php [Accessed: 21 Oct 2013].

Ibm.com. 2013. *UML basics: The class diagram*. [online] Available at: http://www.ibm.com/developerworks/rational/library/content/RationalEdge/sep04/bell/ [Accessed: 24 Oct 2013].

Learncpp.com. 2013. *1.9 — Header files « Learn C++*. [online] Available at: http://www.learncpp.com/cpp-tutorial/19-header-files/ [Accessed: 20 Oct 2013].

Studytonight.com. 2013. *Object Oriented programming Concepts in C++*. [online] Available at: http://www.studytonight.com/cpp/cpp-and-oops-concepts.php [Accessed: 15 Oct 2013].

Tutorialspoint.com. 2013. *C++ Object-Oriented Concepts*. [online] Available at: http://www.tutorialspoint.com/cplusplus/cpp_object_oriented.htm [Accessed: 22 Oct 2013].

Uml-diagrams.org. 2013. *UML Class Diagrams Examples - Abstract Factory Design Pattern,* *Library Management, Online Shopping, Hospital, Digital Imaging in Medicine, Android.*. [online] Available at: http://www.uml-diagrams.org/class-diagrams-examples.html [Accessed: 24 Oct 2013].