Object Oriented Programming

Project: Student Auto Grading System.

Submitted by:

Name: Khushi Shukla

Roll: 19/19

4th semester

CSE DUIET

Contents

Understanding the question

The question is written as follows:

Project: Student Auto Grading System.

Create a class called a Course. Course class will represent a subject taken up by a student. If the student has five courses, then each course is represented by an object of the class Course. Each course offered to the student is unique and can be separated by a course code. Along with this, each course is signified by its name, in-semester mark, and end-semester mark. In addition to these properties, a course is also assigned a credit.

Using the course class discussed above, you will design another two types called Theory Course and Laboratory course (inherit the class Course). A student is declared to pass a theory course if the student secures at least 28 marks in the End semester examination and a total of 40 marks if we add in-semester and end-semester marks together. Similarly, a student is declared to pass a laboratory course if it obtains 35 marks in the end-Semester examination with a total of 50 marks when combined the mark of In-semester and end-semester marks.

In this project, you are going to define all the classes (Course, TheoryCourse, LaboratoryCourse) with the details mentioned above. Along with this, you need to specify the member functions whenever required. In addition, there will be a function common to both Theory and Laboratory courses. That function will calculate the result of the student for that course depends on the constraints mentioned above.

In this project, you also need to define a class called Student. This class will specify the student details like name, roll number, and courses taken up by him/her (both theory and laboratory courses).

In the main function, you should take input for as many students as you want. It should look like as below code:

```
int main()

{

int n;

Sudent *std;

cout&lt;&lt;&quot;\n Enter Number of Student : &quot;;

cin&gt;&gt;n;

std = new Student[n];


for(size_t i; i&lt;n;i++)

std-&gt;details();

//Take the details of student, number of course (create as many course object needed for this

student)

for(size_t i; i&lt;n;i++) {

std-&gt;EndSemestermark(); // Enter all the end semester mark for the student;

std-&gt;InSemestermark(); // Enter all the end Semester mark for the student

}

for(size_t i; i&lt;n;i++) {

//Print result of each course whether Fail/Pass.


}


}
```

To make this project, we first need to define some classes. These classes should fulfill the requirements as written in the question.

Classes required and their details

The classes needed are:

1. Course – represents the course taken by the student.
   Data members:
   a) name of the course
   b) course id
   c) credit
   d) in-semester marks and
   e) end-semester marks in that course

   Member functions:

   a) getCredit() - all the data members are private. So, the getter functions are defined to get the values.
   b) getCourseName() - getter function of course name
   c) getCourseCode() - getter function of course code
   d) viewCourseDeets() – function that displays the details of the student

   A default and a parameterized constructor have also been defined

2. Theory Course – inherits the course class, represents the theory of the course.
   Data members:
   a) in-semester marks (for theory)
   b) end-semester marks and
   c) result

   Member functions:

   a) calcResult() - calculates whether the student has passed the theory or not using the condition mentioned in the question
   b) getResult() - getter function for result
   c) getInsemMarks() - getter function for in semester marks
   d) getEndSemMarks() - getter function for end semester marks
   e) viewResult() - to display the result

   A default and a parameterized constructor have also been defined

3. Laboratory course – inherits the course class, represents the laboratory of the course.
   Data members: same as Theory class

   a) in-semester marks (for lab)
   b) end-semester marks and
   c) result

Member functions: same as theory

   a) calcResult() - calculates whether the student has passed the lab or not using the condition mentioned in the question
   b) getResult() - getter function for result
   c) getInsemMarks() - getter function for in semester marks
   d) getEndSemMarks() - getter function for end semester marks
   e) viewResult() - to display the result

   A default and a parameterized constructor have also been defined

4. Student - specifies the student details like name, roll number, and courses taken up by him/her (both theory and laboratory courses).
   Data members:
   a) Name
   b) Roll
   c) totalCourses – to specify the total number of courses
   d) array of course objects - to store details of the course as a whole
   e) array of theory objects – to store details of theory marks and result
   f) array of lab objects - to store details of lab marks and result
   g) result – stores the result of every student for each course if the students has passed in both lab and theory only then their result is declared as PASS else the result is saved as FAIL.

   Member functions:
   a) details() – this function stores all the detail of the student. This will also save the result of the student for a particular course
   b) showDetails()- to display every detail of the students and courses taken by them
   c) showCourseResult() – to show the result of each course
   d) courseInfo() - getter function for the courses taken by student
   e) labInfo() - getter function for the theory courses taken by student
   f) theoryInfo() - getter function for the lab courses taken by student
   g) getResult() - getter function for the result
   h) getRoll() - getter function for the student's roll
   i) getName() - getter function for the student's name
   j) getTotalCourses() - getter function for the total number of courses

It is important to note that the getter functions are called because the data members are declared privately to achieve data hiding

The main program (Grading.cpp)

Here we have to create an array of student objects and display the details of the students. In addition, these details are saved in a file. In this project for the sake of simplicity, two files have been created. The first file is “Student.csv”; it contains all the details of the student along with the result. The other file is “Result.csv”. this file just stores the student’s name, roll and their result. If they have passed all their courses only then the result is saved as PASS else it is saved as FAIL. **The reason for creating a csv file is that it can be viewed in a tabular format (i.e., Excel). This makes the record look neat and user friendly.** Saving the data in txt file is also possible but the outcome is messy and not what the user would like to work with.

The boiler plate code has already been given in the question. Extra code is written to import and export the data from the file.

Writing in the files

In the main function 2 lines are written which creates 2 new files if they aren’t already created.

```
ofstream fout1("STUDENT.csv");
ofstream fout2("Result.csv");
```

The first file stores every detail of the student and the second file contains only the name, roll and result of the students. This gives the user an option to view either the whole detail which also has the result of students or just the result of students with their roll and name. Then the first row of these files are written i.e., the header is given which will specify the next fields.

```
fout1 << "Roll,Name,CourseCode,CourseName,Credit,Lab-Insem-marks,Lab-Endsem-Marks,Lab-
Result,Theory-Insem-Marks,Theory-Endsem-Marks,Theory-Result,FinalResult";
fout2 << "Roll,Name,Result";
```

To write the details in these files we have to access the data from the particular classes which have that data. Therefore, when the details of the students are being stored at the same time, those details are accessed using various getter functions and these details are written in the file. In Line 102 – line 109 of grading.cpp all the required information is accessed. Now, we have to keep in mind that the string *result of student class stores result for each course separately and user might need the final result. So the final result for every student whether they have passed or not is calculated. For this a Boolean value isPass is declared which is assigned false if any of the course result is fail. After the end of the loop if the isPass variable is true, it means the student has passed every course and hence their final result is written pass in the file otherwise it is Fail.

After all the writing is completed, these files are closed.

```
fout1.close();
fout2.close();
```

Reading from the files

There are two different files and to read from these files two different functions are created.

Read_information()

Ifstream: This data type represents the input file stream and is used to read information from files.

The outermost while loop runs until there is no error found in the stream: `while (fin.good()) {`

`Perform reading }`

Using getline(), file pointer and comma(,) as the delimiter, we read an entire row and store it in a string variable ‘line’: `getline(fin, line, ',');`

The entire line is then separated into words using stringstream() function `stringstream s(line);`

Now, the inner loop iterate over each of these words and print them on the output screen

In this function the setw() function is used which specifies the minimum number of character positions a variable will consume.

Read_result()

This function reads the ‘result.csv’ file and have pretty much the same working as the above function. It is written as a separate function because of two reasons. First is to avoid any confusion and error for future editing of these functions and secondly the spacing needed in both function is not the same. The student.csv file contains 12 fields and result.csv contains 3 fields. So, the second file can have more spacing but if that is given in the first file the output may look very clumsy. Hence, two separate functions are declared.

The outermost while loop runs until there is no error found in the stream: `while (fin.good()) {`

`Perform reading }`

Using getline(), file pointer and comma(,) as the delimiter, we read an entire row and store it in a string variable ‘line’: `getline(fin, line, ',');`

The entire line is then separated into words using stringstream() function `stringstream s(line);`

Now, the inner loop iterate over each of these words and print them on the output screen

The main function

The code for this function has already been provided in the question. However, some extra lines are also added.

At first a pointer to the student class is created to store details of more than one student. Another variable t stores the total number of students. The input and output operations are performed. When we have the value of ‘t’ a loop is made to run that stores the details of each student by calling the member function of student class “details()”. After the details is stored, the whole information for every student is shown in the output.

After the display a while loop will start to run until the user presses ‘0’. It gives three options to the user. Either user can see

1. the course wise result of the student or
2. the final result of the students or
3. all the details of the students.

Depending on the option selected by user, the output is then displayed.

The *Course* Class

This is the base class for both lab and theory class. The data members are

1. courseName string data type – stores the name of the course
2. code of string data type - stores the course code which is unique for each course. To make it unique the first 3 letters of course name and 3 random digits are concatenated.
3. credit of int data type – stores the credit of the course

The in semester and end semester marks are not declared here due to the reason that they will be different for each laboratory and theory exam

A default constructor has been initialized to avoid any warnings and error
The parameterized constructor takes 2 parameters- one for the name of the course and other for the credit of the course - `Course(std::string course_name, int credit_val)`

viewCourseDeets()

This member function is used to display the course details such as the course name course code and course credit. This function is called in the student class. The return type of this function is void.

getCredit()

getCourseName()

getCourseCode()

These are the getter functions that return credit, course name and course code respectively. This getter function is used because all the data members are in private mode to achieve data hiding and hence inaccessible outside the class. The return type of first function is int and for other two is string

The *Theory* Class

This is a derived class of the Course class. The data members present in this class are-

1. insemMarks of data type int - stores the in-semester marks obtained in theory
2. endsemMarks of data type int - stores the end-semester marks obtained in theory
3. result of type string – stores the value ‘PASS’ or ‘FAIL’ depending on the marks

A default constructor have been initialized that sets the insemMarks and endsemMarks as 0 and result as a blank string

The parameterized constructor takes 4 parameters – course_name of type string and insem_marks, endsem_marks and credit_val of type int. the constructor of base class – ‘Course’ is called so that the courseName and credit is initialized and the other 2 data members i.e., insemMarks and endsemMarks are also initialized as given in the argument. The result is initialized as “NIL”

calcResult()

The function has a void return type and is used to set the value of the result data member. The condition is that if

1. marks obtained in end-semester is more than or equal to 35 and
2. the sum of the end-semester and in-semester is more than or equal to 50

only then the result is set as “PASS” else it is set as “FAIL”. Both the condition should be satisfied for the result to be “PASS”

viewResult()

The function has a void return type and displays the end semester and in semester marks and the result

getInsemMarks()

getEndsemMarks()

getResult()

All of these are getter functions for in semester, end semester marks and result respectively. The return type of first 2 is integer and the last one is string

The *Lab* Class

This is a derived class of the Course class and similar to Theory class. However, the difference is that it is for laboratory exams and the condition to calculate result is different. The data members present in this class are-

4. insemMarks of data type int - stores the in-semester marks obtained in theory
5. endsemMarks of data type int - stores the end-semester marks obtained in theory
6. result of type string – stores the value ‘PASS’ or ‘FAIL’ depending on the marks

A default constructor have been initialized that sets the insemMarks and endsemMarks as 0 and result as a blank string

The parameterized constructor takes 4 parameters – course_name of type string and insem_marks, endsem_marks and credit_val of type int. the constructor of base class – ‘Course’ is called so that the courseName and credit is initialized and the other 2 data members i.e., insemMarks and endsemMarks are also initialized as given in the argument. The result is initialized as “NIL”

calcResult()

The function has a void return type and is used to set the value of the result data member. The condition is that if

3. marks obtained in end-semester is more than or equal to 28 and
4. the sum of the end-semester and in-semester is more than or equal to 40

only then the result is set as “PASS” else it is set as “FAIL”. Both the condition should be satisfied for the result to be “PASS”

viewResult()

The function has a void return type and displays the end semester and in semester marks and the result

getInsemMarks()

getEndsemMarks()

getResult()

All of these are getter functions for in semester, end semester marks and result respectively. The return type of first 2 is integer and the last one is string

The *Student* Class

This is the class that makes use of all the other class. The class stores all the detail of the student and displays the result and details in the output.

The data member of this class are:

1. name of string data type – this stores the name of the student.
2. *result of string data type – this stores the results of all the courses of a student
3. roll of int data type – stores the roll number of a student (can be made as a string data type but it is not necessary)
4. totalCourses of int data type – stores the total number of courses taken by each student
5. *course of type Course – this pointer is used to stores the details of all the courses taken up by the student. It can be more than one course.
6. *theorySub of type Theory – this pointer stores the detail of the theory marks of the course.
7. *labSub of type Lab – this pointer stores the detail of the lab marks of the course

A default constructor has been defined to avoid any unwanted warning. It initializes name as an empty string, result as a blank array, totalCourse as 0, and *course, *theorySub and *labSub with size zero.

details()

This function has a void return type and takes the user input and initializes all the values. The first input is taken as name of the student. Here, two lines of code - `cin.ignore(); getline(cin,name);` is used to store the name so that the name can have space in between. Then roll and totalCourses are also taken input by the user. After that the sizes of each of the pointers is declared equal to the totalCourses. A for loop runs for n=totalCourses number of times. i.e., for each course the lab and theory in-semester and end-semester marks and other details such as course name and credit are taken input by the user. Again, the course name is also stored using `cin.ignore(); getline(cin,name);` so that the course name can have more than 1 word.

After the input is taken the course details is stored in the object array of Course class 'course' and the lab and theory marks are stored in the object array of Lab class and Theory class 'labSub' and 'theorySub' respectively.

Then, the lab result is calculated by calling the `calcResult()` of lab class and theory result is calculated by calling the `calcResult()` of theory class. To find the course result to store in *result data member a check is done. If both lab result and theory result is pass the value "PASS" is stored in the *result otherwise "FAIL" is stored.

This member function is called in the main program.

showDetails()

This function has return type of void and used to show every detail of the student. The details shown here are roll, name, all of the courses' name, code, credit, the in-semester marks of lab and theory the end-semester marks of lab and theory. Along with this, the result of lab, the result of theory and the course result as a whole i.e., both lab and theory is displayed.

This member function is also called in main function after the details() member function

showCourseResult()

This function with void return types shows the result of each course for the student. Unlike the above function, no unnecessary detail is shown. The only things displayed in this function is the roll and name of the student and a table having all the courses taken up by that student along with the result.

The setw() used in this function is used to set width which specifies the minimum number of character positions a variable will consume.

This is done so that data is available in a tabular format.

courseInfo()

This function returns a pointer to the class Course. The value to be returned here has the details of all the courses taken up by the student i.e., the data member ‘course’ of Student class. It is obvious that the return type of this function is Course* i.e., a pointer to Course.

labInfo()

This function returns a pointer to the class Lab. The value to be returned here has the details of all the lab marks and result of each course i.e., the data member ‘labSub’ of Student class.

theoryInfo()

The return type of this function is a pointer to the class Theory. The value to be returned here has the details of all the theory marks and result of each course i.e., the data member ‘theorySub’ of Student class.

All these functions given below are getter functions that return the data members of Student class.

getResult()

This function has a return type of string pointer and returns the result of each course taken by the student

getRoll()

It has return type of integer and returns roll number of student.

getName()

getName() has return type string which returns the name of the student

getTotalCourses()

This function has return type of integer and returns the total number of courses the student has taken.

The code

//Course class

```
#include<string>
#pragma once
class Course {
	std::string courseName, code;
	int credit;
public:
	//default constructor defined so that no error or warning is generated
	Course() {
		credit = 0;
		courseName = "";
		int random = 0;
		code = "";
	}

	//parameterized constructor
	Course(std::string course_name, int credit_val) {
		this->credit = credit_val;
		courseName = course_name;
		int random = rand() % 900 + 100; //predefined function to generate random 
numbers between 100 to 999 i.e. 3 digit numbers
		code = courseName.substr(0, 3) + std::to_string(random); //the course code 
is assigned the first 3 letters of the course name and 3 random digits so that it is 
unique
	}


	//to show the details of the course taken by student
	void viewCourseDeets() {
		std::cout << "\n\tCourse code:" << code << "\n\tCourse name : " << 
courseName << "\n\tCredits : " << credit;
	}

	//getter function for credit
	int getCredit() {
		return credit;
	}

	//getter function for course name
	std::string getCourseName() {
		return courseName;
	}

	//getter function for course code
	std::string getCourseCode() {
		return code;
	}
};
```

//Theory class

```
#include"Course.h"
#pragma once
class Theory : public Course {
	int insemMarks, endsemMarks;
	std::string result;
public:
	//default constructor defined so that no error or warning is generated
	Theory() {
		insemMarks = 0;
		endsemMarks = 0;
		result = "";
	}

	//parameterized constructor
	Theory(int insem_marks, int endsem_marks, std::string course_name, int credit_val)
: Course( course_name, credit_val){
		insemMarks = insem_marks;
		endsemMarks = endsem_marks;
		result = "NIL";
	}

	//calculates whether the student has passed the theory or not using the condition
mentioned in the question
	void calcResult() {
		if (endsemMarks >= 28 && (insemMarks + endsemMarks >= 40))
			this->result="PASS";
		else
			this->result="FAIL";
	}

	//getter function for result
	std::string getResult() {
		return this->result;
	}

	//to display the result
	void viewResult() {
		std::cout << "\n\tTHEORY:\n" << "\t\tInsem Marks : " << insemMarks <<
"\n\t\tEndsem Marks : " << endsemMarks << "\n\t\tResult : " << result;
	}

	//getter function for insem marks
	int getInsemMarks() {
		return insemMarks;
	}

	//getter function for end sem marks
	int getEndSemMarks() {
		return endsemMarks;
	}
};
```

//Lab class

```
#include"Course.h"
#pragma once
class Lab : public Course {
	int insemMarks, endsemMarks;
	std::string result;
public:
	//default constructor defined so that no error or warning is generated
	Lab() {
		insemMarks = 0;
		endsemMarks = 0;
		result = "";
	}

	//parameterized constructor
	Lab(int insem_marks, int endsem_marks, std::string course_name, int credit_val):
Course(course_name, credit_val) {
		insemMarks = insem_marks;
		endsemMarks = endsem_marks;
		result = "NIL";
	}


	//calculates whether the student has passed the theory or not using the condition
mentioned in the question
	void calcResult() {
		if (endsemMarks >= 35 && (endsemMarks + insemMarks >= 50))
			this->result="PASS";
		else
			this->result="FAIL";
	}

	//getter function for result
	std::string getResult() {
		return result;
	}

	//to display the result
	void viewResult() {
		std::cout << "\n\tLAB:\n" << "\t\tInsem Marks : " << insemMarks <<
"\n\t\tEndsem Marks : " << endsemMarks<<"\n\t\tResult : "<<result ;
	}

	//getter function for insem marks
	int getInsemMarks() {
		return insemMarks;
	}

	//getter function for end sem marks
	int getEndSemMarks() {
		return endsemMarks;
	}
};
```

//Student class

```
#include <iomanip>
#include"Course.h"
#include"Lab.h"
#include"Theory.h"
using namespace std;

#pragma once
```

```
class Student {
       std::string name, *result;
       int roll, totalCourses;
       Course* course;
       Theory* theorySub;
       Lab* labSub;
public:
       Student() {
              name = "";
              result = {};
              roll = 0;
              totalCourses = 0;
              course = new Course[0];
              theorySub = new Theory[0];
              labSub = new Lab[0];
       }
       void details() {

              cout << "\nEnter the student's name: ";
              cin.ignore(); //  to ignore or clear one or more characters from the input
buffer. we need to clear the input buffer, otherwise it will occupy the buffer of
previous variable. By pressing the "Enter" key after the first input, as the buffer of
previous variable has space to hold new data, the program skips the following input of
container.
              getline(cin,name); //to accept name with spaces

              cout << "\nEnter roll number of the student(only integer value): ";
              cin >> roll;

              cout << "\nEnter total courses enrolled: ";
              cin >> totalCourses;

              //declaring size of the data members
              result = new std::string[totalCourses];
              course = new Course[totalCourses];
              labSub = new Lab[totalCourses];
              theorySub = new Theory[totalCourses];

              for (int i = 0; i < totalCourses; i++) {
                     std::string cname; //temporary variables
                     int insemLab, endsemLab, insemTheory, endsemTheory, credit;
//temporary variables

                     //taking user input
                     cout << "\n\tEnter course name " << i + 1 << ": ";
                     cin.ignore(); //  to ignore or clear one or more characters from the
input buffer. we need to clear the input buffer, otherwise it will occupy the buffer of
previous variable. By pressing the "Enter" key after the first input, as the buffer of
previous variable has space to hold new data, the program skips the following input of
container.
                     getline(cin, cname); //to accept name with spaces

                     cout << "\n\tEnter credit of " << cname << " : ";
                     cin >> credit;
                     cout << "\n\tLAB:\n";
                     cout << "\n\t\tEnter insem marks obtained in " << cname << " lab :
";

                     cin >> insemLab;
```

```
                        cout << "\n\t\tEnter endsem marks obtained in " << cname << " lab : 
";
                        cin >> endsemLab;

                        cout << "\n\tTHEORY:\n";
                        cout << "\n\t\tEnter insem marks obtained in " << cname << " theory 
: ";
                        cin >> insemTheory;
                        cout << "\n\t\tEnter endsem marks obtained in " << cname << " theory 
: ";
                        cin >> endsemTheory;

                        //storing the course detail
                        course[i] = Course(cname, credit);


                        //storing the lab detials
                        labSub[i] = Lab(insemLab, endsemLab, cname, credit);

                        //storing the theory details
                        theorySub[i] = Theory(insemTheory, endsemTheory, cname, credit);

                        //calculating lab result
                        labSub[i].calcResult();

                        //calculating theory result
                        theorySub[i].calcResult();

                        //if the students has passed in both lab and theory only then their 
result is declared as PASS else the result is saved as FAIL.
                        if (labSub[i].getResult() == "PASS" && theorySub[i].getResult() == 
"PASS") {
                               this->result[i] = "PASS";
                        }
                        else
                               this->result[i] = "FAIL";


                 }
          }

          //display every detail of the students and courses taken by them
          void showDetails() {
                 cout << "\nRoll: " << roll << " Name: " << name << "\n" << "Courses 
enrolled: ";
                 for (int i = 0; i < totalCourses; i++) {
                        cout << "\n\t" << i + 1 << ")";
                        course[i].viewCourseDeets();
                        labSub[i].viewResult();
                        theorySub[i].viewResult();
                        cout << "\nCOURSE RESULT : " << this->result[i] << "\n";

                 }
          }

          //to show course result
          void showCourseResult() {
                 cout << "\nRoll: " << roll << " Name: " << name << "\n";
```

```
		cout << "Course name" << setw(29) << "Result\n";
		for (int i = 0; i < totalCourses; i++) {
			//cout << "\n" << i + 1 << ")";
			cout << course[i].getCourseName()<<setw(34)<< this->result[i];
			cout << "\n";
		}
		cout << "\n";
	}

	//getter function for the courses taken by student
	Course* courseInfo() {
		return course;
	}

	//getter function for the theory courses taken by student
	Lab* labInfo() {
		return labSub;
	}

	//getter function for the lab courses taken by student
	Theory* theoryInfo() {
		return theorySub;
	}


	//getter function for the result
	std::string* getResult() {
		return result;
	}

	//getter function for the student's roll
	int getRoll() {
		return roll;
	}

	//getter function for the student's name
	std::string getName() {
		return name;
	}


	//getter function for the total number of courses
	int getTotalCourses() {
		return totalCourses;
	}

};
```

//the main program - Grading.cpp

```
#include <iostream>
#include<fstream>
#include <iomanip>
#include <sstream>
#include"Student.h"
#include"Course.h"
#include"Theory.h"
```

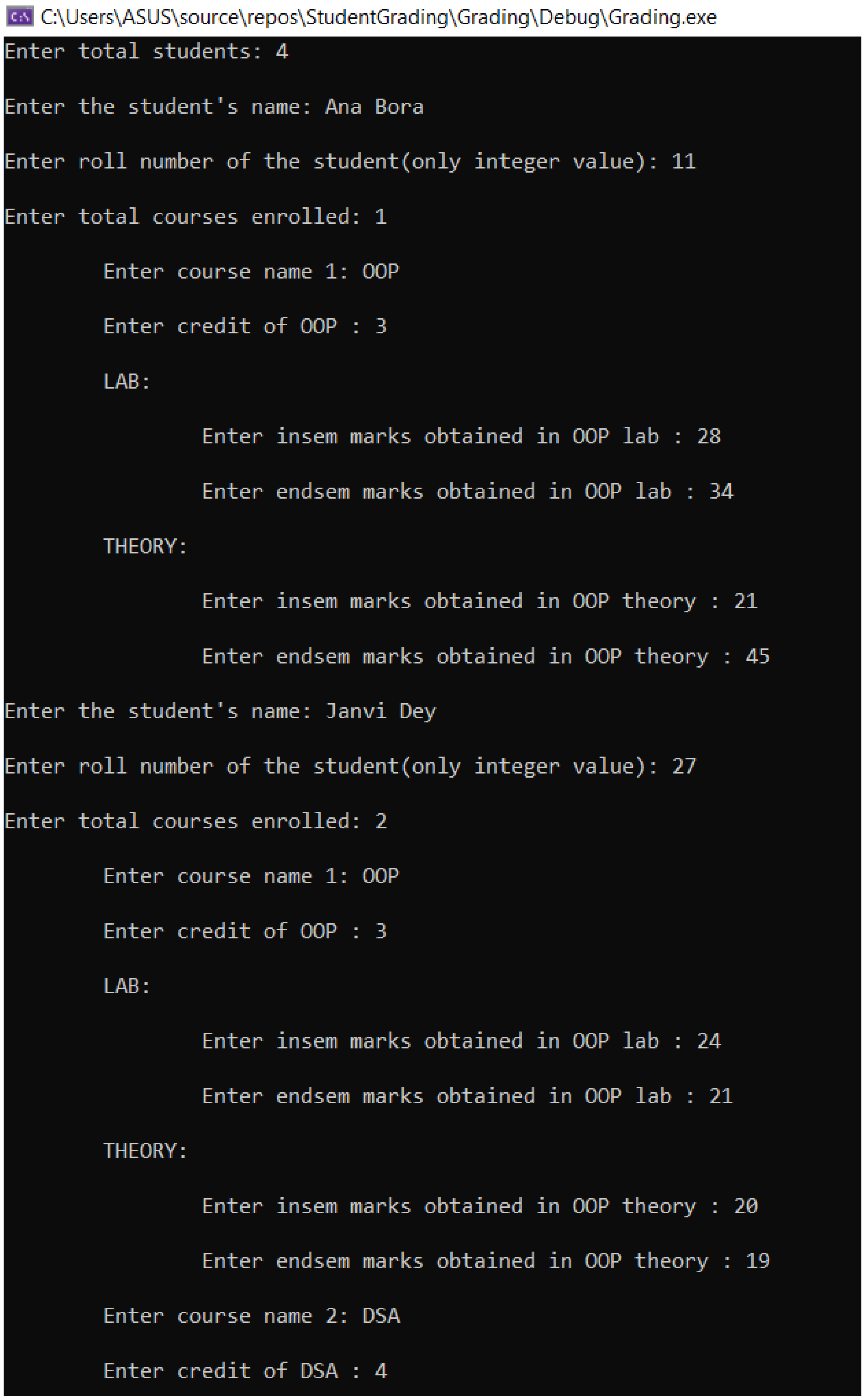
```cpp
#include"Lab.h"
using namespace std;

void read_information() {

    //to read the file
    ifstream fin;

    // Open an existing file
    fin.open("STUDENT.csv");

    string line, word;
    while (fin.good()) {

        //seperating the lines at commas (,)
        getline(fin, line, ',');

        //breaking the line in words
        stringstream s(line);

        //displaying each word

        while (getline(s, word, ',')) {
            cout << left << setw(17) << word;  //setw() is used to set width
        }

    }

    cout << "\n";
    cout << "The data may appear a little distorted due to less space. Check out
Student.csv file to see the details clearly\n";

}

void read_result() {

    //to read the file
    ifstream fin;

    // Open an existing file
    fin.open("Result.csv");

    string line, word;
    while (fin.good()) {
        //seperating the lines at commas (,)
        getline(fin, line, ',');

        //breaking the line in words
        stringstream s(line);

        //displaying each word
        while (getline(s, word, ',')) {
            cout<<left << setw(20) << word;
        }

    }
    cout << "\n";
    cout << "The data may appear a little distorted due to less space. Check out 
Student.csv file to see the details clearly\n";

}

int main()
{

    Student* s;
    int t, isTrue=0;
    std::cout << "Enter total students: ";
    cin >> t;
    s = new Student[t];


    //storing in the csv file writing in the file. if the file is not present, a new one 
of the given names are created
    ofstream fout1("STUDENT.csv");
    ofstream fout2("Result.csv");

    //writing the headers of the csv file
    fout1 << "Roll,Name,CourseCode,CourseName,Credit,Lab-Insem-marks,Lab-Endsem-
Marks,Lab-Result,Theory-Insem-Marks,Theory-Endsem-Marks,Theory-Result,FinalResult";
    fout2 << "Roll,Name,Result";



    for (int i = 0; i < t; i++) {
        //storing the values by calling a member function
        s[i].details();


        //accessing all the values to store in csv file
        int roll = s[i].getRoll();
        string name = s[i].getName();
        int totalCourse = s[i].getTotalCourses();
        string* result = s[i].getResult();
        Course* course = s[i].courseInfo();
        Lab* lab = s[i].labInfo();
        Theory* theory = s[i].theoryInfo();

        fout1 << "\n" << roll << "," << name << ",";
        fout2 << "\n" << roll << "," << name << ",";

        // to check if a student has passed all the exams
        bool isPass = true;

        for (int j = 0; j < totalCourse; j++) {

            //writing in the csv file
            if (result[j] == "FAIL")
                isPass = false; //if student fails in any one subject then they fail the 
exam as a whole

            if (j != 0)
                fout1 << "\n,,";
            fout1 << course[j].getCourseCode() << "," << course[j].getCourseName() << "," 
<< course[j].getCredit() << "," << lab[j].getInsemMarks()
```

```
                << "," << lab[j].getEndSemMarks() << "," << lab[j].getResult() << "," <<
theory[j].getInsemMarks() << ","
                << theory[j].getEndSemMarks() << "," << theory[j].getResult() << "," <<
result[j];
        }

        //checking if student is  pass or not
        if (isPass)
            fout2 << "PASS";
        else
            fout2 << "FAIL";

    }
    //end of the loop

    fout1.close(); //close the file student.csv
    fout2.close(); //close the file result.csv


    for (int i = 0; i < t; i++)
        s[i].showDetails(); //display the result along with the details


    //display statement
    std::cout << "\nThe information has been saved in the student.csv file\nThe result of
the students is saved in Result.csv file\n";

    //a loop so that user can see both the reult and the details of the student as
required
    while (1) {
        std::cout << "Press 1 to view the course wise result\nPress 2 to view the overall
result of each student.\nPress 3 to view the full information.\nPress 0 to break\n----";
        cin >> isTrue;
        cout << "\n";

        if (isTrue == 0) {
            break;
        }
        if (isTrue == 1) {
            for (int i = 0; i < t; i++) {
                s[i].showCourseResult();
                cout << "\n";
            }
        }
        else if (isTrue == 2) {
            read_result();
            cout << "\n";
        }
        else if (isTrue == 3) {
            read_information();
            cout << "\n";
        }

        else {
            cout << "Wrong option! Try again\n";
        }
    }
    return 0;
```

```
}
```

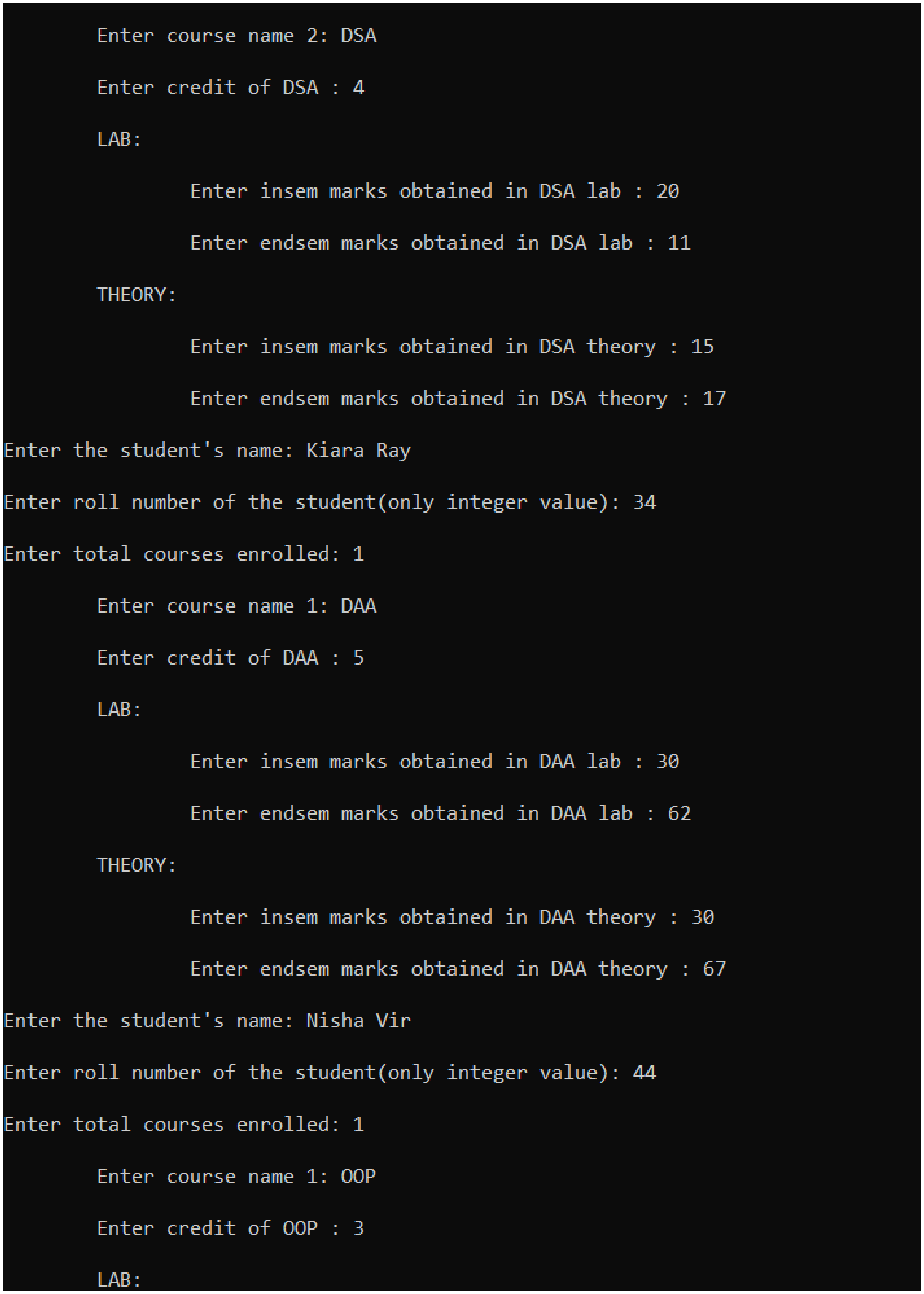
The output

Output of the code

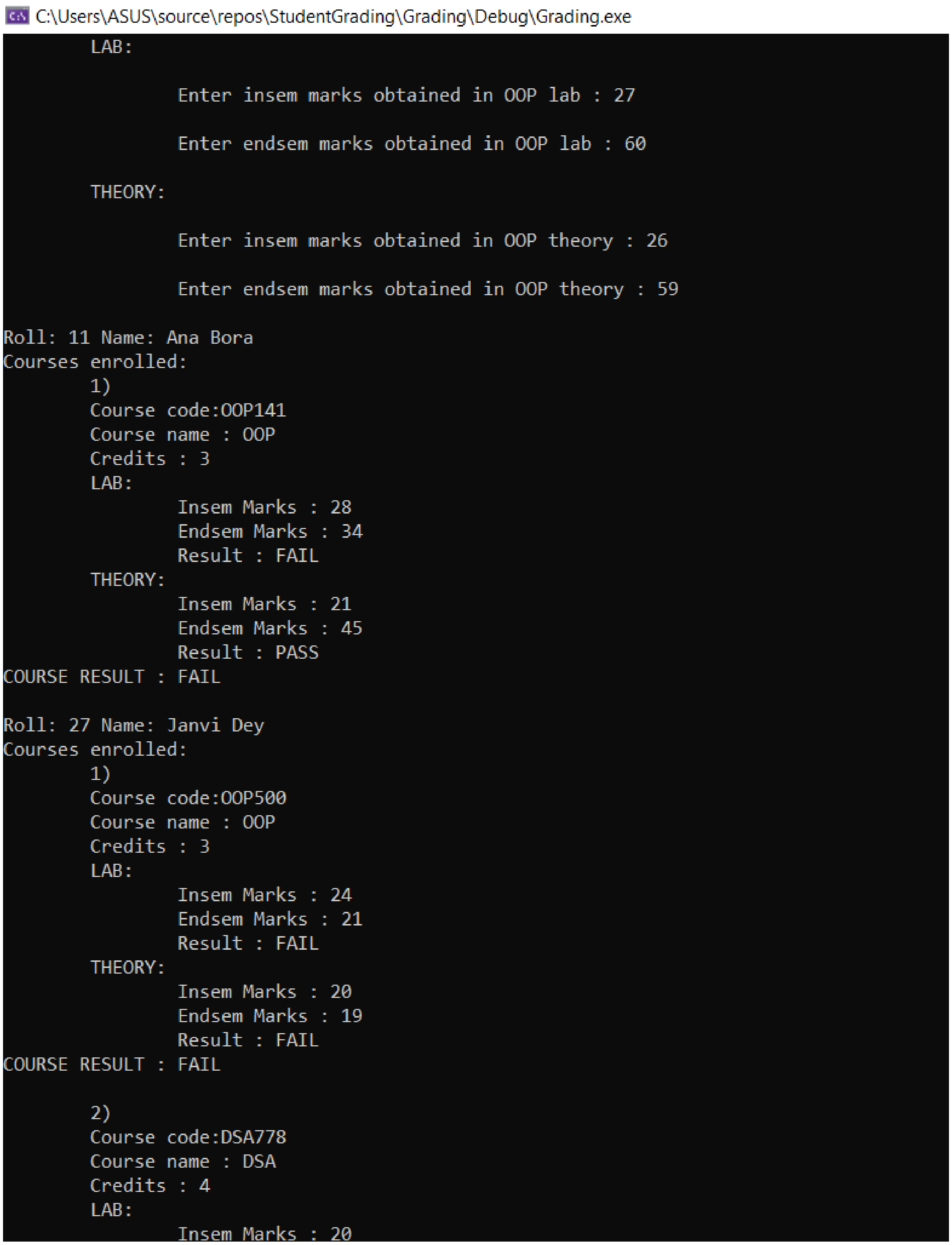


C:\Users\ASUS\source\repos\StudentGrading\Grading\Debug\Grading.exe

```
Enter total students: 4

Enter the student's name: Ana Bora

Enter roll number of the student(only integer value): 11

Enter total courses enrolled: 1

        Enter course name 1: OOP

        Enter credit of OOP : 3

        LAB:

                Enter insem marks obtained in OOP lab : 28

                Enter endsem marks obtained in OOP lab : 34

        THEORY:

                Enter insem marks obtained in OOP theory : 21

                Enter endsem marks obtained in OOP theory : 45

Enter the student's name: Janvi Dey

Enter roll number of the student(only integer value): 27

Enter total courses enrolled: 2

        Enter course name 1: OOP

        Enter credit of OOP : 3

        LAB:

                Enter insem marks obtained in OOP lab : 24

                Enter endsem marks obtained in OOP lab : 21

        THEORY:

                Enter insem marks obtained in OOP theory : 20

                Enter endsem marks obtained in OOP theory : 19

        Enter course name 2: DSA

        Enter credit of DSA : 4
```

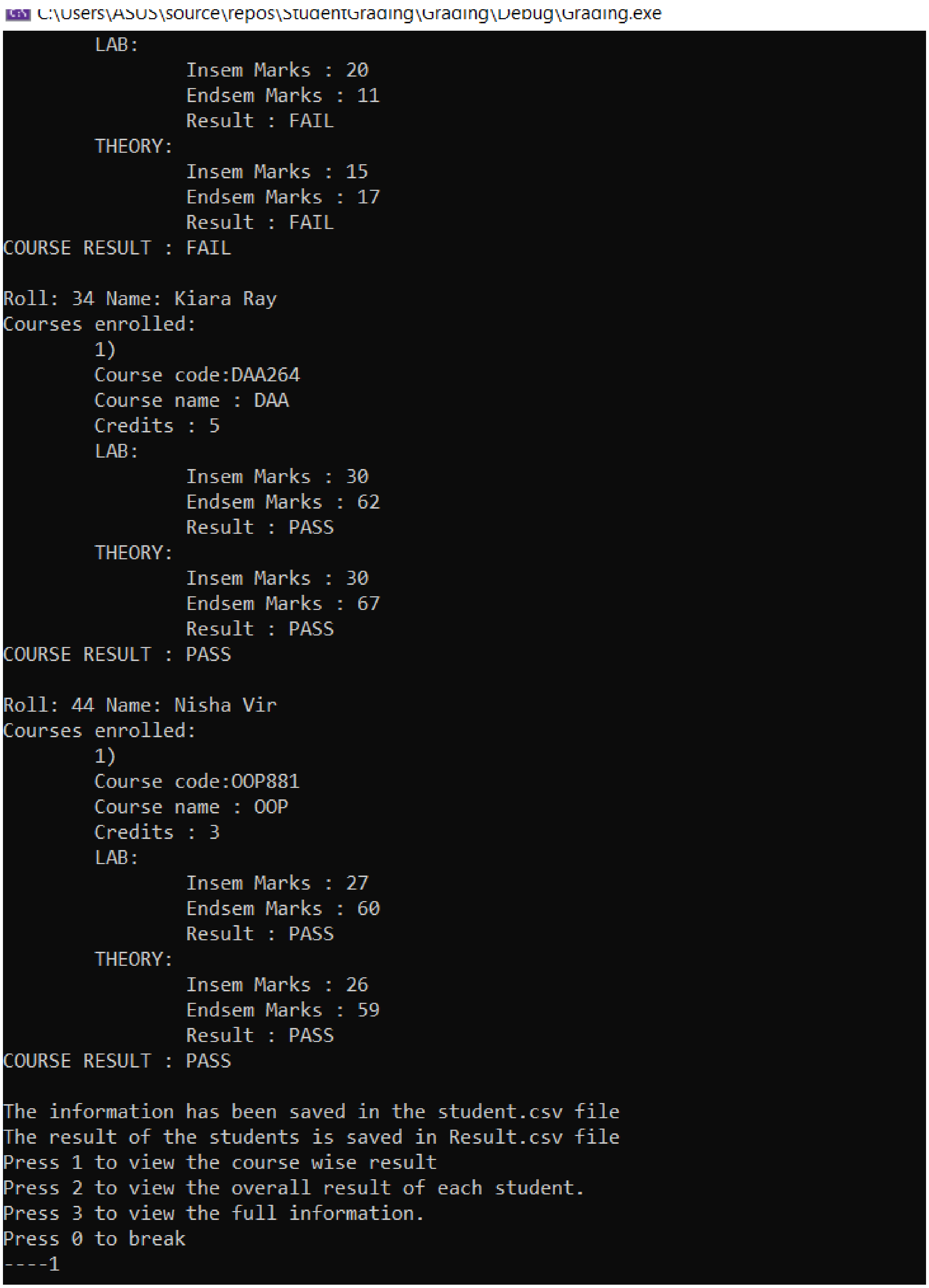


```
        Enter course name 2: DSA

        Enter credit of DSA : 4

        LAB:

                Enter insem marks obtained in DSA lab : 20

                Enter endsem marks obtained in DSA lab : 11

        THEORY:

                Enter insem marks obtained in DSA theory : 15

                Enter endsem marks obtained in DSA theory : 17

Enter the student's name: Kiara Ray

Enter roll number of the student(only integer value): 34

Enter total courses enrolled: 1

        Enter course name 1: DAA

        Enter credit of DAA : 5

        LAB:

                Enter insem marks obtained in DAA lab : 30

                Enter endsem marks obtained in DAA lab : 62

        THEORY:

                Enter insem marks obtained in DAA theory : 30

                Enter endsem marks obtained in DAA theory : 67

Enter the student's name: Nisha Vir

Enter roll number of the student(only integer value): 44

Enter total courses enrolled: 1

        Enter course name 1: OOP

        Enter credit of OOP : 3

        LAB:
```



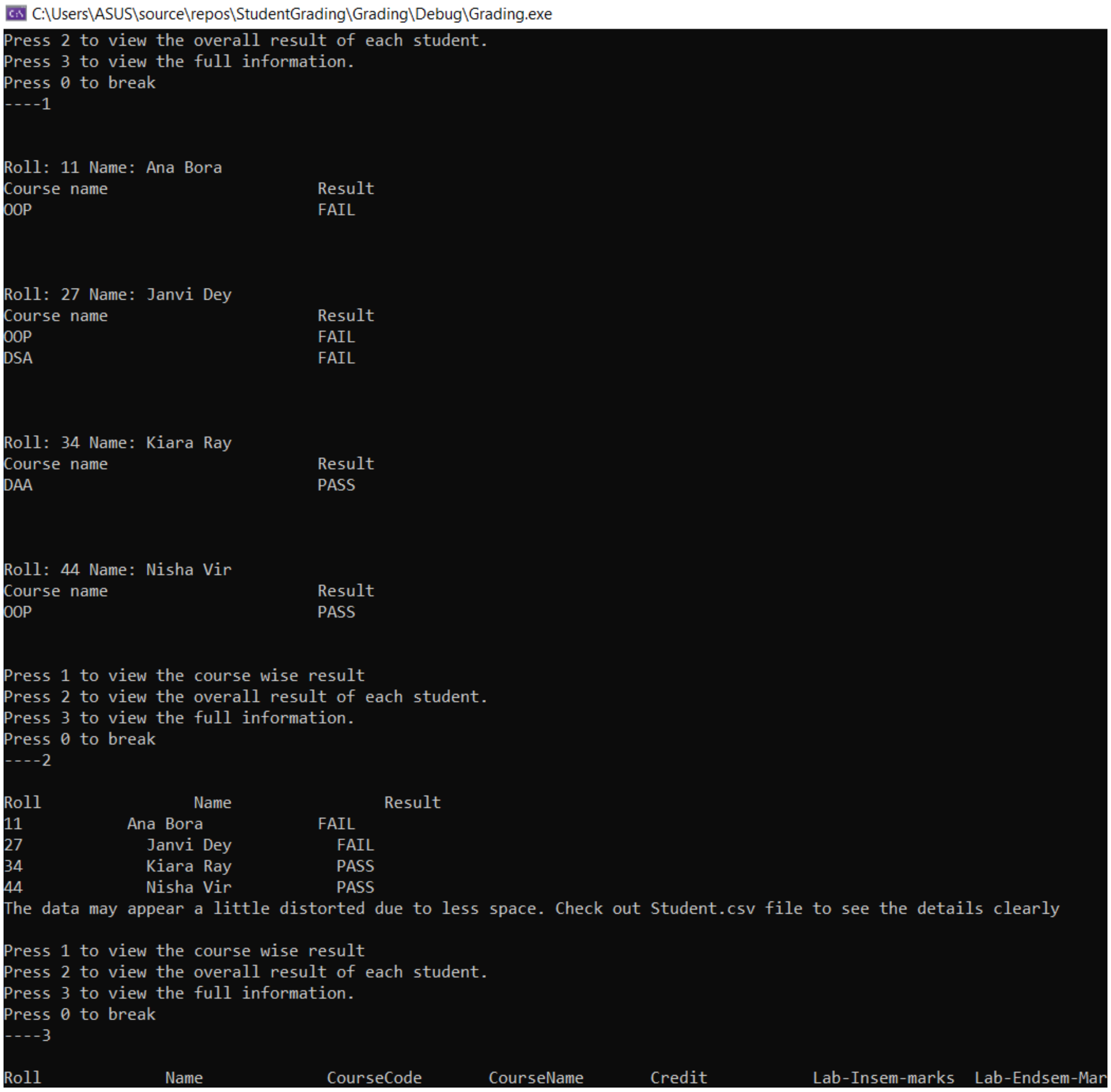
C:\Users\ASUS\source\repos\StudentGrading\Grading\Debug\Grading.exe

```
        LAB:

                Enter insem marks obtained in OOP lab : 27

                Enter endsem marks obtained in OOP lab : 60

        THEORY:

                Enter insem marks obtained in OOP theory : 26

                Enter endsem marks obtained in OOP theory : 59

Roll: 11 Name: Ana Bora
Courses enrolled:
        1)
        Course code:OOP141
        Course name : OOP
        Credits : 3
        LAB:
                Insem Marks : 28
                Endsem Marks : 34
                Result : FAIL
        THEORY:
                Insem Marks : 21
                Endsem Marks : 45
                Result : PASS
COURSE RESULT : FAIL

Roll: 27 Name: Janvi Dey
Courses enrolled:
        1)
        Course code:OOP500
        Course name : OOP
        Credits : 3
        LAB:
                Insem Marks : 24
                Endsem Marks : 21
                Result : FAIL
        THEORY:
                Insem Marks : 20
                Endsem Marks : 19
                Result : FAIL
COURSE RESULT : FAIL

        2)
        Course code:DSA778
        Course name : DSA
        Credits : 4
        LAB:
                Insem Marks : 20
```

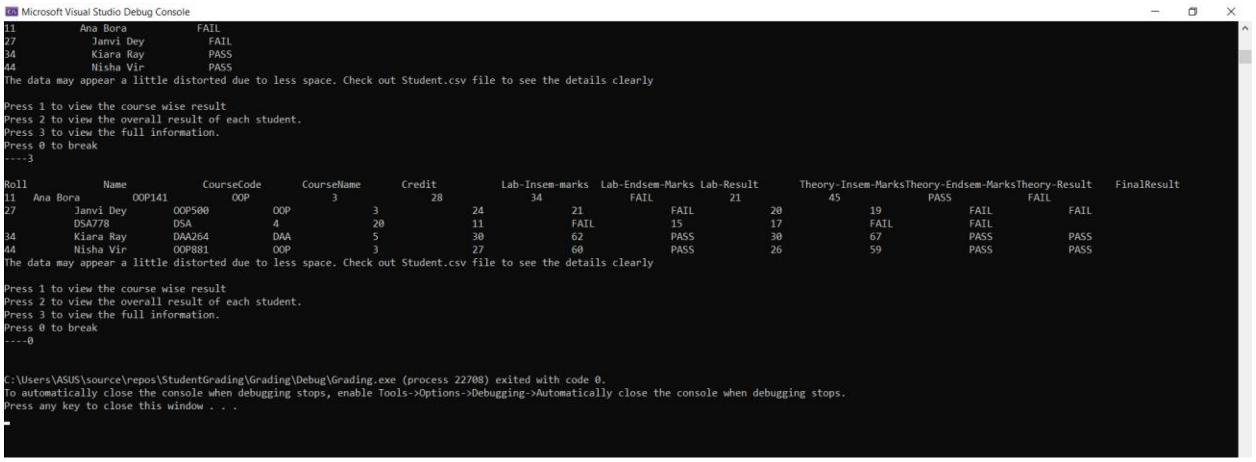


C:\Users\ASUS\source\repos\StudentGrading\Grading\Debug\Grading.exe

```
        LAB:
                Insem Marks : 20
                Endsem Marks : 11
                Result : FAIL
        THEORY:
                Insem Marks : 15
                Endsem Marks : 17
                Result : FAIL
COURSE RESULT : FAIL

Roll: 34 Name: Kiara Ray
Courses enrolled:
        1)
        Course code:DAA264
        Course name : DAA
        Credits : 5
        LAB:
                Insem Marks : 30
                Endsem Marks : 62
                Result : PASS
        THEORY:
                Insem Marks : 30
                Endsem Marks : 67
                Result : PASS
COURSE RESULT : PASS

Roll: 44 Name: Nisha Vir
Courses enrolled:
        1)
        Course code:OOP881
        Course name : OOP
        Credits : 3
        LAB:
                Insem Marks : 27
                Endsem Marks : 60
                Result : PASS
        THEORY:
                Insem Marks : 26
                Endsem Marks : 59
                Result : PASS
COURSE RESULT : PASS

The information has been saved in the student.csv file
The result of the students is saved in Result.csv file
Press 1 to view the course wise result
Press 2 to view the overall result of each student.
Press 3 to view the full information.
Press 0 to break
----1
```


C:\Users\ASUS\source\repos\StudentGrading\Grading\Debug\Grading.exe

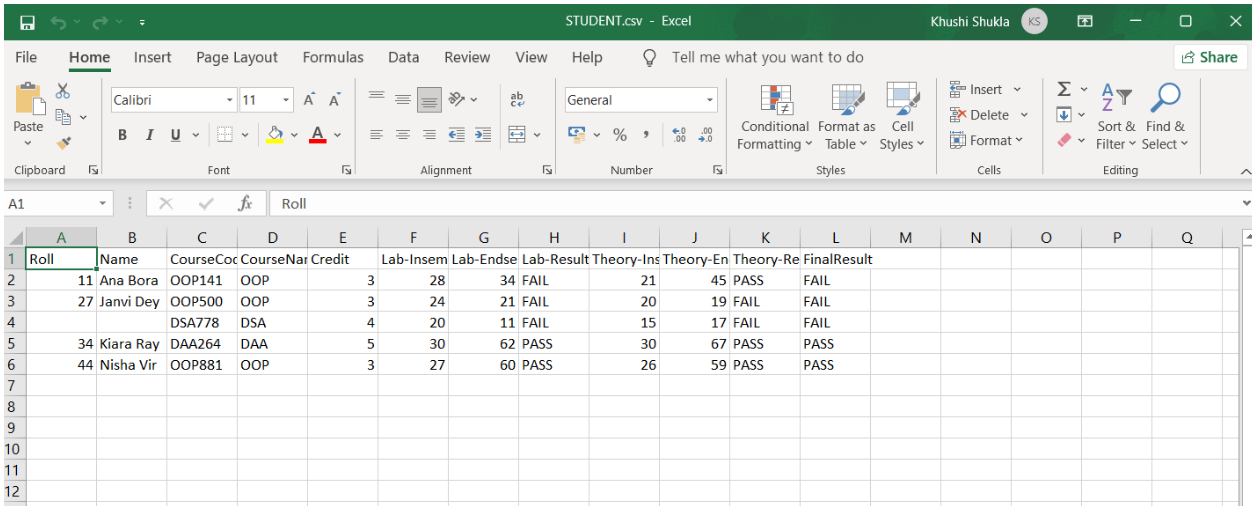
```
Press 2 to view the overall result of each student.
Press 3 to view the full information.
Press 0 to break
----1


Roll: 11 Name: Ana Bora
Course name                      Result
OOP                              FAIL



Roll: 27 Name: Janvi Dey
Course name                      Result
OOP                              FAIL
DSA                              FAIL



Roll: 34 Name: Kiara Ray
Course name                      Result
DAA                              PASS



Roll: 44 Name: Nisha Vir
Course name                      Result
OOP                              PASS


Press 1 to view the course wise result
Press 2 to view the overall result of each student.
Press 3 to view the full information.
Press 0 to break
----2

Roll                Name                Result
11           Ana Bora            FAIL
27             Janvi Dey           FAIL
34             Kiara Ray           PASS
44             Nisha Vir           PASS
The data may appear a little distorted due to less space. Check out Student.csv file to see the details clearly

Press 1 to view the course wise result
Press 2 to view the overall result of each student.
Press 3 to view the full information.
Press 0 to break
----3

Roll                 Name                CourseCode       CourseName       Credit           Lab-Insem-marks  Lab-Endsem-Mar
```


Microsoft Visual Studio Debug Console

```
11          Ana Bora          FAIL
27            Janvi Dey         FAIL
34            Kiara Ray         PASS
44            Nisha Vir         PASS
The data may appear a little distorted due to less space. Check out Student.csv file to see the details clearly

Press 1 to view the course wise result
Press 2 to view the overall result of each student.
Press 3 to view the full information.
Press 0 to break
----3

Roll                 Name               CourseCode        CourseName        Credit            Lab-Insem-marks  Lab-Endsem-Marks Lab-Result        Theory-Insem-MarksTheory-Endsem-MarksTheory-Result    FinalResult
11   Ana Bora         OOP141            OOP               3                 28                34                FAIL              21                45                PASS              FAIL
27            Janvi Dey        OOP500            OOP               3                 24                21                FAIL              20                19                FAIL              FAIL
              DSA778           DSA               4                 20                11                FAIL              15                17                FAIL              FAIL
34            Kiara Ray        DAA264            DAA               5                 30                62                PASS              30                67                PASS              PASS
44            Nisha Vir        OOP881            OOP               3                 27                60                PASS              26                59                PASS              PASS
The data may appear a little distorted due to less space. Check out Student.csv file to see the details clearly

Press 1 to view the course wise result
Press 2 to view the overall result of each student.
Press 3 to view the full information.
Press 0 to break
----0


C:\Users\ASUS\source\repos\StudentGrading\Grading\Debug\Grading.exe (process 22708) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

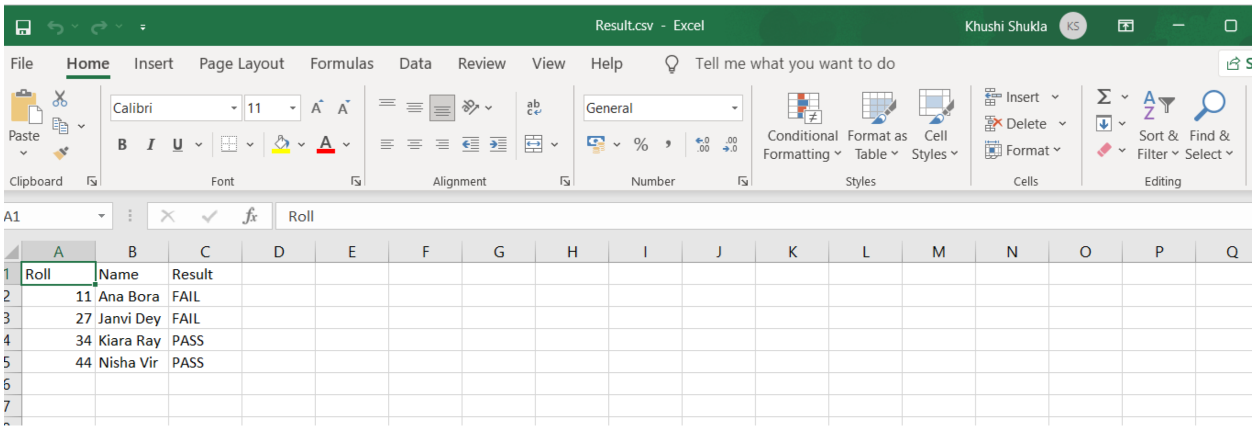
Output of the csv files

Student.csv



Roll	Name	CourseCoc	CourseNar	Credit	Lab-Insem	Lab-Endse	Lab-Result	Theory-Ins	Theory-En	Theory-Re	FinalResult
11	Ana Bora	OOP141	OOP	3	28	34	FAIL	21	45	PASS	FAIL
27	Janvi Dey	OOP500	OOP	3	24	21	FAIL	20	19	FAIL	FAIL
		DSA778	DSA	4	20	11	FAIL	15	17	FAIL	FAIL
34	Kiara Ray	DAA264	DAA	5	30	62	PASS	30	67	PASS	PASS
44	Nisha Vir	OOP881	OOP	3	27	60	PASS	26	59	PASS	PASS

Result.csv



Roll	Name	Result
11	Ana Bora	FAIL
27	Janvi Dey	FAIL
34	Kiara Ray	PASS
44	Nisha Vir	PASS