# WORKSHEET 2

**Submitted By: Sandhya Aryal**

**Student ID: 24000860**

**Cyber Security and Digital Forensics**

GitHub Link:

https://github.com/Sandhyaaaa1/Cpp_Worksheet

**Task 1: Basic student grading system prototype using classes and objects.**

**[30 Marks]**

Write a program that manages a simple student grade calculator with the following requirements. Create a Student class that has:

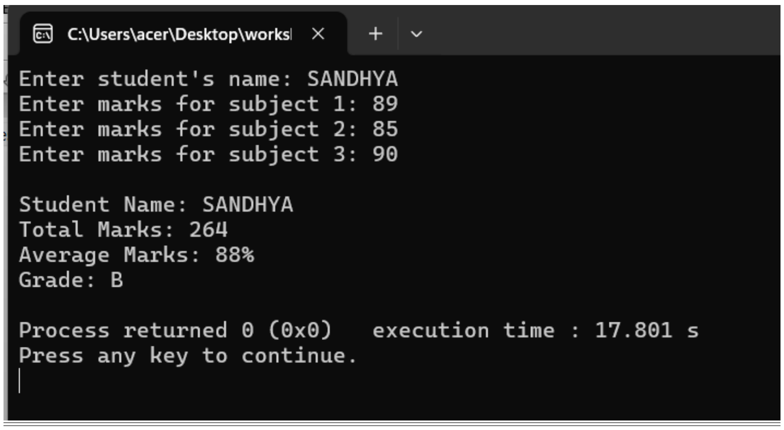
1. Student name (string)
2. Three subject marks (integers)
3. A basic member function to calculate average

```
#include <iostream>

#include <string>

using namespace std;

class Student

{

private:

   string name;

   int marks[3];

public:
  void input_Details()

{

  cout << "Enter the name of students: ";

  cin >> name;

  for (int i = 0; i < 3; i++)

    {

      cout << "Enter marks of subject " << i + 1 << ": ";

      cin >> marks[i];



      while (marks[i] < 0 || marks[i] > 100)

    {

      cout << "Your input is Invalid! The marks must be between 0 and 100. Please re-enter: ";

      cin >> marks[i];

    }
  }
}
```

```
int calculate_Total()

  {

      return marks[0] + marks[1] + marks[2];

  }

double calculate_Average()

  {

      return static_cast<double>(calculate_Total()) / 3;



  }

char calculate_Grade()

  {
      double average = calculate_Average();

      if (average >= 90)

        {
            return 'A';

        }

  else if (average >= 80)
```

```
        {
            return 'B';

        }

    else if (average >= 70)

        {

            return 'C';

        }

    else if (average >= 60)

        {

            return 'D';
        }


    else

        {

            return 'F';
    }
}
```

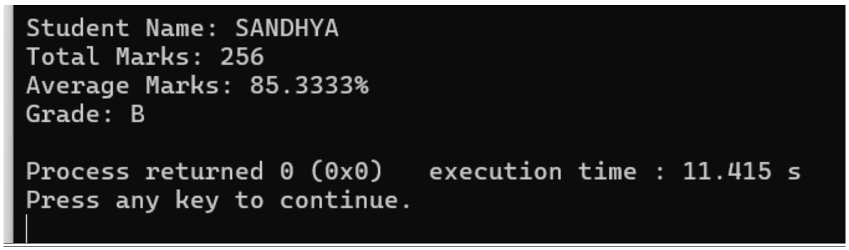
```
    void display_Results()

    {

        cout << "\nStudent Name: " << name << endl;

        cout << "Total Marks: " << calculate_Total() << endl;

        cout << "Average Marks: " << calculate_Average() << "%" << endl;

        cout << "Grade: " << calculate_Grade() << endl;

    }
};

void Student_Grading()

{
    Student s1;

    s1.input_Details();

    s1.display_Results();

}

int main()

{
    Student_Grading();
    return 0;
}
```



```
Enter student's name: SANDHYA
Enter marks for subject 1: 89
Enter marks for subject 2: 85
Enter marks for subject 3: 90

Student Name: SANDHYA
Total Marks: 264
Average Marks: 88%
Grade: B

Process returned 0 (0x0)   execution time : 17.801 s
Press any key to continue.
```

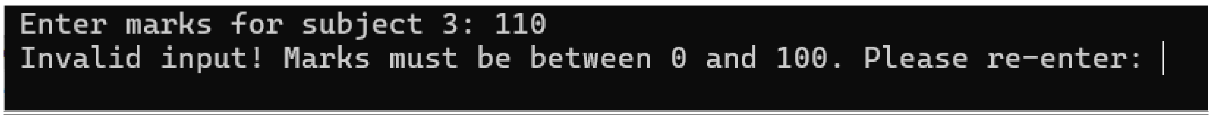
The program should:

1. Accept student details (name and marks) from user input
2. Calculate and display:
    1. Total marks
    2. Average marks
    3. Grade (A for ≥90%, B for ≥80%, C for ≥70%, D for ≥60%, F for <60%)



```
Student Name: SANDHYA
Total Marks: 256
Average Marks: 85.3333%
Grade: B

Process returned 0 (0x0)   execution time : 11.415 s
Press any key to continue.
```

3. Display a message if any mark is below 0 or above 100



```
Enter marks for subject 3: 110
Invalid input! Marks must be between 0 and 100. Please re-enter:
```

## Task 2: Programming assignments: All questions are mandatory

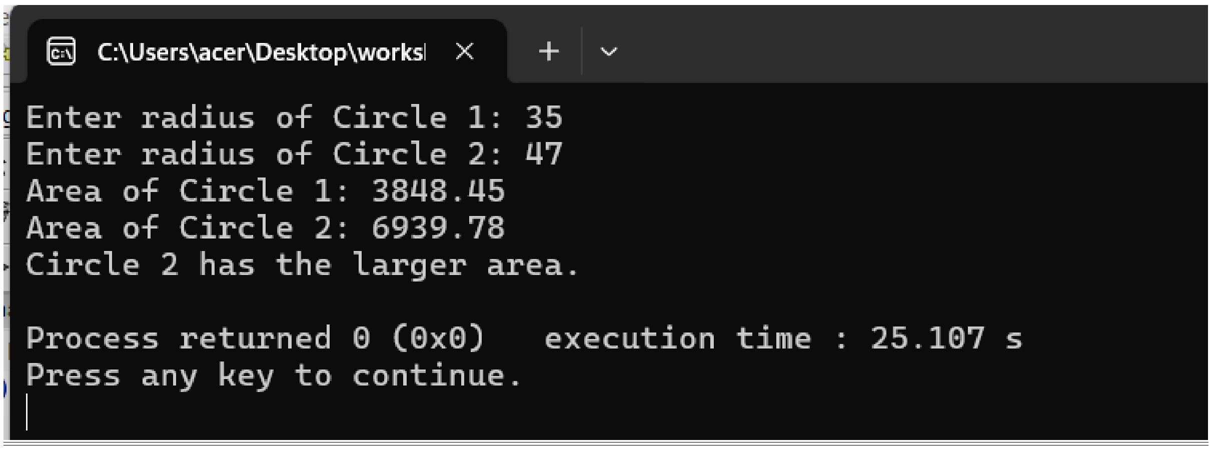
1. **Write a program with a class Circle having:**
    1. **Private member: radius (float)**
    2. **A constructor to initialize radius**
    3. **A friend function compareTwoCircles that takes two Circle objects and prints which circle has the larger area**

```
#include <iostream>

#include <cmath>

using namespace std;



class Circle;


void compareTwoCircles(Circle &c1, Circle &c2);


class Circle


{

private:

  float radius;


public:


  Circle(float r)

  {
      radius = r;
  }



  float area()

  {
      return M_PI * radius * radius;
  }



  friend void compareTwoCircles(Circle &c1, Circle &c2);

};



void compareTwoCircles(Circle &c1, Circle &c2)

{
  float area1 = c1.area();

  float area2 = c2.area();

  cout << "Area of Circle 1: " << area1 << endl;

  cout << "Area of Circle 2: " << area2 << endl;



  if (area1 > area2)
```

```
      {

      cout << "Circle 1 has the larger area." << endl;
      }
   else if (area1 < area2)
      {
      cout << "Circle 2 has the larger area." << endl;
      }
   else
      {
      cout << "Both circles have the same area." << endl;
      }
}

int main()
{
   float radius1, radius2;

   cout << "Enter radius of Circle 1: ";

   cin >> radius1;

   cout << "Enter radius of Circle 2: ";

   cin >> radius2;

   Circle circle1(radius1);

   Circle circle2(radius2);

   compareTwoCircles(circle1, circle2);
```

```
    return 0;
}
```

**OUTPUT:**



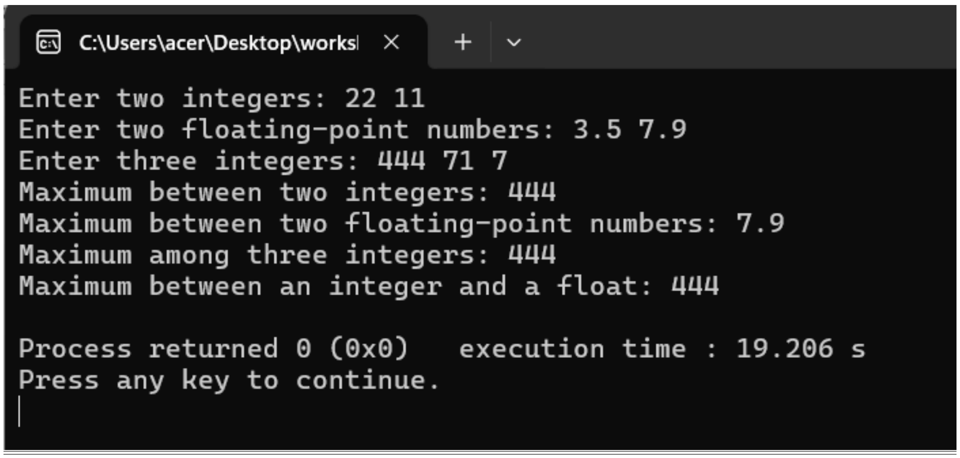
```
Enter radius of Circle 1: 35
Enter radius of Circle 2: 47
Area of Circle 1: 3848.45
Area of Circle 2: 6939.78
Circle 2 has the larger area.

Process returned 0 (0x0)   execution time : 25.107 s
Press any key to continue.
```

2. **Create a program with these overloaded functions named findMax:**
    1. **One that finds maximum between two integers**
    2. **One that finds maximum between two floating-point numbers**
    3. **One that finds maximum among three integers**
    4. **One that finds maximum between an integer and a float**

```
#include <iostream>
using namespace std;

class Maximum
```

```
{

public:


    int findMax(int a, int b)

    {

        return (a > b) ? a : b;

    }

    float findMax(float a, float b)

    {

        return (a > b) ? a : b;

    }


    int findMax(int a, int b, int c)

    {

        return (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);

    }
    float findMax(int a, float b)

    {

        return (a > b) ? a : b;

    }
};

int main()

{
    Maximum maximum;

    int int1, int2, int3;

    float float1, float2;



    cout << "Enter two integers: ";

    cin >> int1 >> int2;



    cout << "Enter two floating-point numbers: ";

    cin >> float1 >> float2;



    cout << "Enter three integers: ";
    cin >> int1 >> int2 >> int3;

    cout << "Maximum between two integers: " << maximum.findMax(int1, int2) << endl;

    cout << "Maximum between two floating-point numbers: " << maximum.findMax(float1, float2) << endl;

    cout << "Maximum among three integers: " << maximum.findMax(int1, int2, int3) << endl;

    cout << "Maximum between an integer and a float: " << maximum.findMax(int1, float1) << endl;

    return 0;
}
```

**OUTPUT:**



```
Enter two integers: 22 11
Enter two floating-point numbers: 3.5 7.9
Enter three integers: 444 71 7
Maximum between two integers: 444
Maximum between two floating-point numbers: 7.9
Maximum among three integers: 444
Maximum between an integer and a float: 444

Process returned 0 (0x0)   execution time : 19.206 s
Press any key to continue.
```

## Task 3: Basics of File Handling

**Write a program that reads the titles of 10 books (use an array of 150 characters) and writes them in a binary file selected by the user. The program should read a title and display a message to indicate if it is contained in the file or not.**

```
#include <iostream>

#include <fstream>
```

```
#include <string>

#include <vector>

#include <limits>

#include <stdexcept>

using namespace std;

const int MAX_BOOKS = 10;

const int MAX_TITLE_LENGTH = 150;



void writeBookTitles(const string& filename, char books[][MAX_TITLE_LENGTH])

{

  ofstream outFile(filename, ios::binary);

  if (!outFile)

    {

      cout << "Error opening file for writing.\n";

      return;
    }
```

```
   for (int i = 0; i < MAX_BOOKS; ++i)

      {

         outFile.write(books[i], MAX_TITLE_LENGTH);
      }

      outFile.close();
}

bool searchBookTitle(const string& filename, const string& title)

   {


      ifstream inFile(filename, ios::binary);

      if (!inFile)

         {
            cout << "Error opening file for reading.\n";

            return false;
         }

      char buffer[MAX_TITLE_LENGTH];

      while (inFile.read(buffer, MAX_TITLE_LENGTH))

         {
```

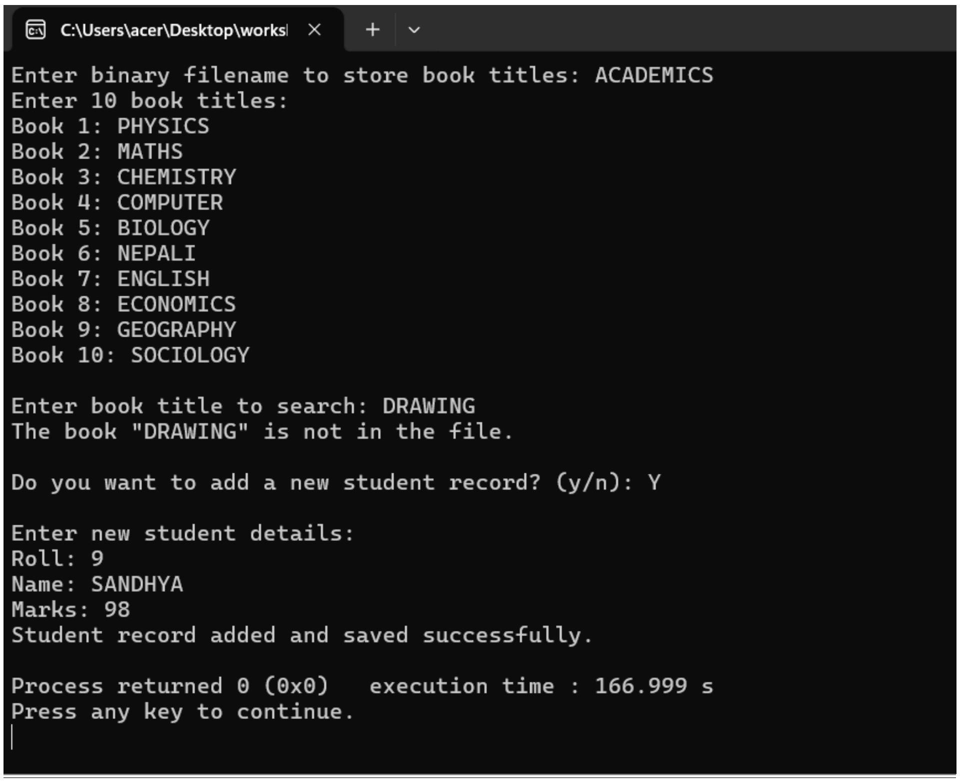
```
        if (title == buffer)

          {

              return true;
      }
   }

   inFile.close();

   return false;
}

struct Student
{

   int roll;

   string name;

   float marks;

};

void readStudentsFromFile(const string& filename, vector<Student>& students)

{
    ifstream inFile(filename);

    if (!inFile) {

        cout << "Student file not found. A new one will be created.\n";

        return;
    }

    Student s;

    while (inFile >> s.roll >> ws && getline(inFile, s.name, ',') && inFile >> s.marks) {

        if (s.marks < 0 || s.marks > 100)

            throw out_of_range("Invalid marks found in file.");

        students.push_back(s);
    }

    inFile.close();
}

void addStudentRecord(vector<Student>& students) {

    Student s;

    cout << "\nEnter new student details:\n";

    cout << "Roll: ";
    cin >> s.roll;

    cin.ignore();

    cout << "Name: ";

    getline(cin, s.name);

    cout << "Marks: ";

    cin >> s.marks;



    if (s.marks < 0 || s.marks > 100)

        throw out_of_range("Marks must be between 0 and 100.");

    students.push_back(s);
}



void saveStudentsToFile(const string& filename, const vector<Student>& students)

{



    ofstream outFile(filename);

    if (!outFile)

        {
```

```
        cerr << "Failed to save students.\n";

        return;
    }

    for (const auto& s : students)

        {

        outFile << s.roll << " " << s.name << "," << s.marks << endl;
    }

    outFile.close();
}


int main()

{

    char books[MAX_BOOKS][MAX_TITLE_LENGTH];

    string bookFile;

    cout << "Enter binary filename to store book titles: ";

    getline(cin, bookFile);

    cout << "Enter 10 book titles:\n";
```

```
for (int i = 0; i < MAX_BOOKS; ++i)

   {

   cout << "Book " << i + 1 << ": ";

   cin.getline(books[i], MAX_TITLE_LENGTH);
}

writeBookTitles(bookFile, books);


string searchTitle;

cout << "\nEnter book title to search: ";

getline(cin, searchTitle);

if (searchBookTitle(bookFile, searchTitle))


   cout << "The book \"" << searchTitle << "\" is in the file.\n";

else

   cout << "The book \"" << searchTitle << "\" is not in the file.\n";


vector<Student> students;

string studentFile = "students.txt";
```

```
try
 {
   readStudentsFromFile(studentFile, students);
}
 catch (const exception& e)
 {
   cerr << "Exception while reading students: " << e.what() << endl;
}
char choice;
cout << "\nDo you want to add a new student record? (y/n): ";
cin >> choice;
if (choice == 'y' || choice == 'Y') {
   try
   {
      addStudentRecord(students);
      saveStudentsToFile(studentFile, students);
```

```
        cout << "Student record added and saved successfully.\n";

      }

      catch (const exception& e) {

        cerr << "Error adding student: " << e.what() << endl;
      }
    }

    return 0;
}
```



```
Enter binary filename to store book titles: ACADEMICS
Enter 10 book titles:
Book 1: PHYSICS
Book 2: MATHS
Book 3: CHEMISTRY
Book 4: COMPUTER
Book 5: BIOLOGY
Book 6: NEPALI
Book 7: ENGLISH
Book 8: ECONOMICS
Book 9: GEOGRAPHY
Book 10: SOCIOLOGY

Enter book title to search: DRAWING
The book "DRAWING" is not in the file.

Do you want to add a new student record? (y/n): Y

Enter new student details:
Roll: 9
Name: SANDHYA
Marks: 98
Student record added and saved successfully.

Process returned 0 (0x0)   execution time : 166.999 s
Press any key to continue.
```

**Create a program that:**

1. **Reads student records (roll, name, marks) from a text file**
2. **Throws an exception if marks are not between 0 and 100**

3. **Allows adding new records with proper validation**

4. **Saves modified records back to file**

```
#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <stdexcept>

using namespace std;

class Student

{

private:

   int roll;

   string name;

   int marks;

public:


   Student(int r, const string& n, int m) : roll(r), name(n), marks(m) {}
```

```
int getRoll() const
{
   return roll;
}
string getName() const
{
   return name;
}
int getMarks() const
{
   return marks;
}
static void validateMarks(int marks)
{
   if (marks < 0 || marks > 100)
```

```
        {
            throw out_of_range("Marks must be between 0 and 100.");
        }
    }

    void display() const

        {
            cout << "Roll Number: " << roll << ", Name: " << name << ", Marks: " << marks << endl;
        }
};

class StudentManager

    {
private:

    vector<Student> students;

    string filename;

public:


    StudentManager(const string& file) : filename(file)

        {
            readStudentRecords();
        }

        void readStudentRecords()
```

```
  {

    ifstream file(filename);

if (!file)

  {

    cerr << "Error opening file for reading." << endl;
    return;
  }


int roll, marks;


string name;

while (file >> roll)

  {
    file.ignore();

    getline(file, name);

    file>> marks;

    file.ignore();

    students.push_back(Student(roll, name, marks));
```

```
        }

           file.close();
}


void addStudentRecord()

{
   int roll, marks;

   string name;

   cout << "Enter student roll number: ";

   cin >> roll;

   cin.ignore();

   cout << "Enter student name: ";

   getline(cin, name);

   cout << "Enter student marks: ";

   cin >> marks;

   try

   {
      Student::validateMarks(marks);
```

```
        students.push_back(Student(roll, name, marks));

        cout << "New student record added successfully!" << endl;
    }

    catch (const out_of_range& e)

    {

        cout << "Error: " << e.what() << endl;

    }

}

void displayStudentRecords() const

{

    if (students.empty())

            {

                cout << "No records available." << endl;
                return;
            }

                cout << "\nStudent Records:\n";
```

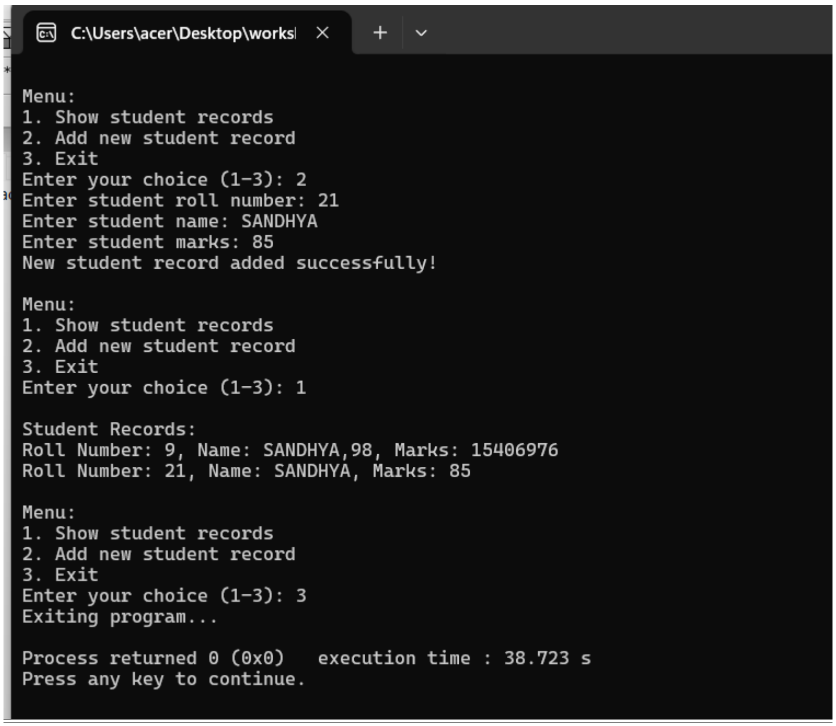
```
      for (const auto& student : students)

        {

          student.display();

        }
}

void saveStudentRecords() const

  {
      ofstream file(filename);

      if (!file)

      {

        cerr << "Error opening file for writing." << endl;

        return;
  }

  for (const auto& student : students)

      {

        file << student.getRoll() << endl;

        file << student.getName() << endl;
```

```
            file << student.getMarks() << endl;
          }

            file.close();
    }
};

int main()

    {

        string filename = "students.txt";

        StudentManager manager(filename);

    int choice;
    bool running = true;

    while (running)

        {

        cout << "\nMenu:\n";

        cout << "1. Show student records\n";

        cout << "2. Add new student record\n";

        cout << "3. Exit\n";

        cout << "Enter your choice (1-3): ";
```

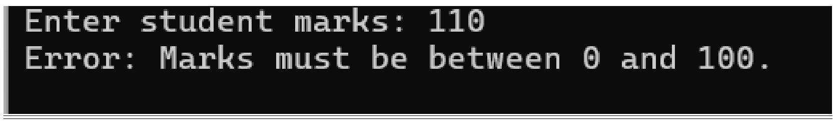
```
cin >> choice;

switch (choice)

{
  case 1:

    manager.displayStudentRecords();

    break;

  case 2:

    manager.addStudentRecord();

    break;

  case 3:


    manager.saveStudentRecords();

    cout << "Exiting program...\n";

    running = false;

    break;

  default:
```

```
            cout << "Invalid choice, please try again.\n";

            break;
        }
    }

    return 0;
}
```

**OUTPUT:**



```
Menu:
1. Show student records
2. Add new student record
3. Exit
Enter your choice (1-3): 2
Enter student roll number: 21
Enter student name: SANDHYA
Enter student marks: 85
New student record added successfully!

Menu:
1. Show student records
2. Add new student record
3. Exit
Enter your choice (1-3): 1

Student Records:
Roll Number: 9, Name: SANDHYA,98, Marks: 15406976
Roll Number: 21, Name: SANDHYA, Marks: 85

Menu:
1. Show student records
2. Add new student record
3. Exit
Enter your choice (1-3): 3
Exiting program...

Process returned 0 (0x0)   execution time : 38.723 s
Press any key to continue.
```



```
Enter student marks: 110
Error: Marks must be between 0 and 100.
```