Worksheet – 2

Name : Crishtina K.C.

Student ID : 23085130

Cyber Security And Digital Forensics

Github link:

https://github.com/crishtina01/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

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

```
#include <iostream>

#include <string>

using namespace std;

class Student
```

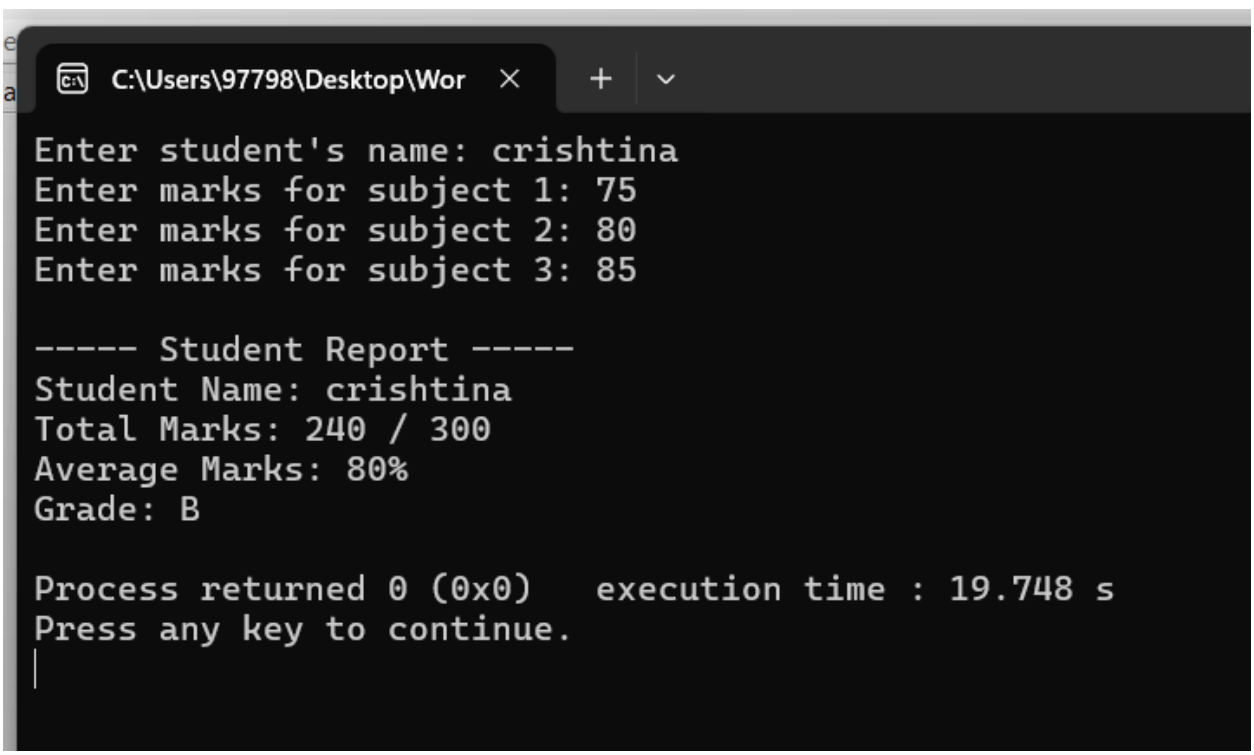
```
{

private:

   string name;

   int marks[3];

public:

   void inputData()

      {

         cout << "Enter student's name: ";

         cin >> name;

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

      {

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

         cin >> marks[i];
```

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

   {

         cout << "Input Invalid! Marks must be between 0 and 100. 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 displayResults()

    {
        cout << "\n----- Student Report -----" << endl;

        cout << "Student Name: " << name << endl;

        cout << "Total Marks: " << calculate_Total() << " / 300" << endl;

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

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

    }
};

void StudentGradingsystem()

{
    Student s1;
```

```
    s1.inputData();

    s1.displayResults();
}

int main()

{
    StudentGradingsystem();

    return 0;

}
```



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**

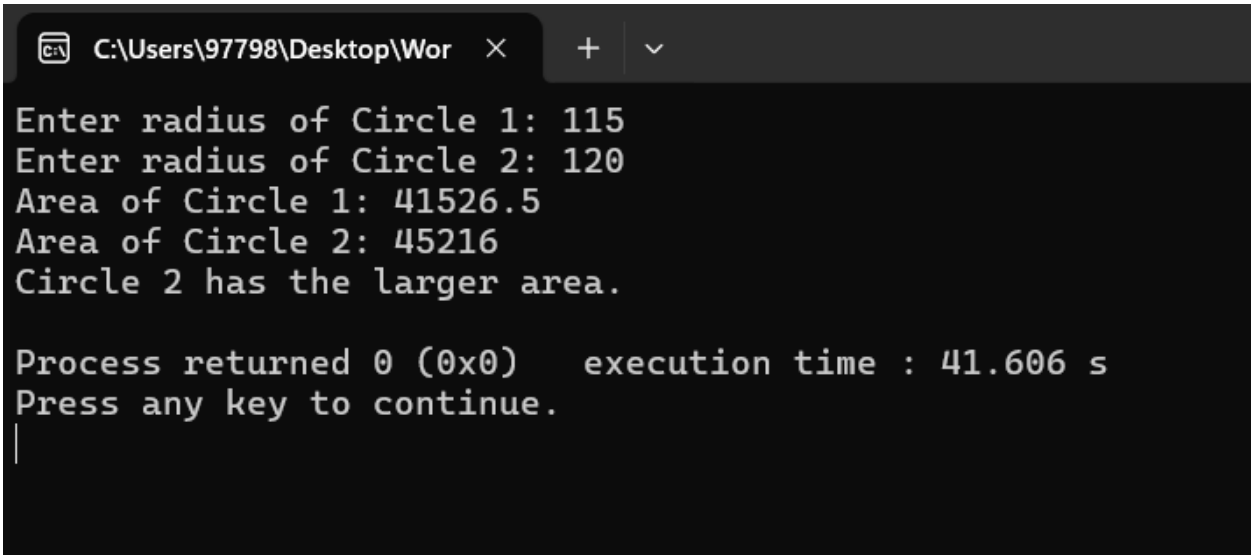
```
#include <iostream>

#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 3.14 * 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 r1, r2;

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

  cin >> r1;

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

  cin >> r2;
```

```
    Circle circle1(r1);

    Circle circle2(r2);

    compareTwoCircles(circle1, circle2);

    return 0;
}
```



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**

One that finds maximum between an integer and a float

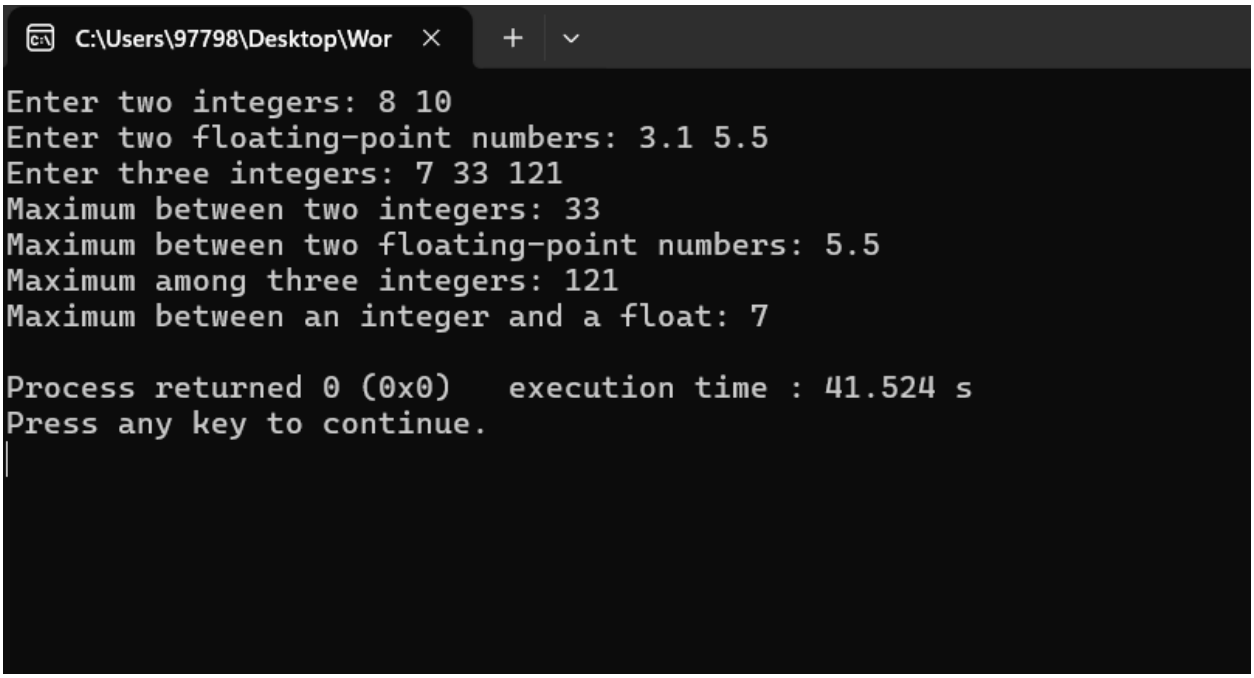
```
#include <iostream>

using namespace std;
```

```
class Max_Finder

{

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()

  {
    Max_Finder max_Finder;

    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: " << max_Finder.findMax(int1, int2) <<
endl;

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

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

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

```
  return 0;
}
```



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 <cstring>

using namespace std;
```

```
const int MAX_TITLE_LENGTH = 150;

const int NUM_BOOKS = 10;


void writeBooksToFile(const string& filename, const string books[])

{

   ofstream outFile(filename, ios::binary);

   if (!outFile)

   {
      cout << "Error opening your file for writing." << endl;

      return;
   }

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

      {
         int length = books[i].length();

         outFile.write(reinterpret_cast<char*>(&length), sizeof(length));
```

```
        outFile.write(books[i].c_str(), length);
    }

    outFile.close();
}


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

{

    ifstream inFile(filename, ios::binary);


    if (!inFile)

      {

        cout << "Error opening your file for reading." << endl;

        return false;
    }

    bool found = false;
    int length;
```

```
    char buffer[MAX_TITLE_LENGTH];

    while (inFile.read(reinterpret_cast<char*>(&length), sizeof(length)))

      {
        inFile.read(buffer, length);

        buffer[length] = '\0'; // Null terminate the string

      if (title == buffer)

        {
          found = true;

          break;
      }
    }

    inFile.close();
    return found;
}

int main()

{
    string books[NUM_BOOKS];
```

```
cout << "Enter 10 book titles: " << endl;

for (int i = 0; i < NUM_BOOKS; i++) {

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

   getline(cin, books[i]);

}


string filename = "books.dat";

writeBooksToFile(filename, books);


string searchTitle;

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

getline(cin, searchTitle);

if (searchBookInFile(filename, searchTitle))
```

```
    {

    cout << "The book \"" << searchTitle << "\" is in the file." << endl;
  }
  else

    {

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

  return 0;
}
```

```
C:\Users\97798\Desktop\Wor

Enter 10 book titles:
Book 1: physics
Book 2: chemistry
Book 3: maths
Book 4: english
Book 5: it ends with you
Book 6: computer
Book 7: biology
Book 8: nepali
Book 9: it starts with you
Book 10: geography

Enter the book title to search for: it ends with you
The book "it ends with you" is in the file.

Process returned 0 (0x0)   execution time : 86.900 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)

        {

            cout << "Error opening your 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 << "Successfully added new students record!" << 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)
```

```
        {

            cout << "Error opening your 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;
}
```