Batch: B2	Roll No.: 16010124107
Experiment / assignment / tutorial No.	

TITLE : Student Result Processing System Using Classes in C++

AIM: To implement a student result processing system in C++ using classes and objects, focusing on encapsulation, constructors, and member functions

---

Expected OUTCOME of Experiment:

CO1:Apply the features of object oriented programming languages. (C++ and Java)
CO2:Explore arrays, vectors, classes and objects in C++ and Java

---

Books/ Journals/ Websites referred:

1. E. Balagurusamy, “Programming with Java”, McGraw-Hill.
2. E. Balagurusamy, “Object Oriented Programming with C++”, McGraw-Hill.

Design a C++ program that creates and manages records of students using classes. Your program must:

1. **Store data for multiple students.**
2. **Accept marks for 3 subjects.**
3. **Calculate total and average marks.**
4. **Display details of all students.**
5. **Identify the student with the highest average.**

Each student should have:

- **Roll Number (int)**

- **Name (string)**
- **Marks in 3 subjects (float)**
- **Total (float)**
- **Average (float)**

Requirements:

- **Create a class `Student` with private data members.**
- **Use public member functions to:**
  - **Accept input (with a constructor or separate `input()` function).**
  - **Calculate total and average.**
  - **Display individual student data.**
- **Maintain multiple student records using an array of objects.**

Variations/Modifications:

1. Modify your class to include student grades (A/B/C) based on average marks.
2. Write a function to sort the students by total marks in descending order.
3. Rewrite the program using dynamic memory allocation instead of a fixed-size array.
4. Add a static data member to count how many Student objects were created.
5. Add a search function to find and display the record of a student by roll number.

Pre Lab/ Prior Concepts:

1. Basic Structure of a C++ Program

- Understanding the syntax of a C++ program: `#include`, `main()` function, return types.
- Use of headers like `<iostream>` and `using namespace std`.

2. Input and Output Operations

- Use of `cin` and `cout` for reading user input and displaying output.
- Formatting output using manipulators like `setw`, `fixed`, `setprecision` (if included).

3. Variables and Data Types

- Declaring variables of type `int`, `float`, `char`, and `string`.
- Understanding literals, constants, and data ranges.

4. Conditional Statements

- Use of `if`, `if-else`, and `switch` statements to perform decision-making based on input data.

5. Loops

- Implementation of `for`, `while`, and `do-while` loops.
- Use cases: repeating input operations, iterating over arrays or records.

6. Arrays

- Declaring and using single-dimensional arrays.
- Storing and accessing data for multiple entities like marks of students.

7. Functions

- Writing reusable blocks of code using functions.
- Function declaration, definition, calling, and parameter passing (by value/reference).

8. Introduction to Classes in C++

- Definition and declaration of a class.
- Understanding the syntax for:
    - **Private** and **Public** access specifiers.
    - **Member variables** and **member functions**.
- Creating **objects** and accessing class members.

9. Constructors (Optional for your experiment)

- Understanding how constructors are used to initialize object data automatically.
- Syntax of default and parameterized constructors.

10. Arrays of Objects

- Using an array to store multiple objects of a class (e.g., `Student s[50];`).
- Accessing member functions for each object inside a loop.

Algorithm:

1. Start
2. declare private data members
3. declare public data members
4. declare public methods
5. populate function bodies using scope resolution
6. create main function
7. input number of students
8. create vector of objects and populate it using a loop
9. in the for loop, keep checking for highest average and store that index.
10. display details of students
11. display highest average student in the end
12. stop

Implementation details:

```
#include <bits/stdc++.h>

using namespace std;


class Student{

   private:

   int roll;

   string name;

   int marks[3];


   public:
```

```
    float total, average;

    public:
    void input();
    void display();
    void calculate();
};

void Student::input(){
    cout << "Enter roll number and name.\n";
    cin >> roll >> name;
    cout << "Enter marks for 3 subjects\n";
    for(int i=0;i<3;i++)
    {
        cin >> marks[i];
    }
}


void Student::calculate(){
    total=0;
    for(int i=0;i<3;i++)
    {
        total +=marks[i];
```

```
    }
    average = total/3.0f;
}

void Student::display()
{
    cout << "Name of the student: " << name << endl;
    cout << "Roll number of the student: " << roll << endl;
    cout << "Total marks of the student: " << total << endl;
    cout << "Average of the student: " << average << endl;
}

int main()
{
    cout << "Enter number of students\n";
    int n; cin >> n;
    vector<Student>v(n);
    float highest=0;
    int indexoftopper=0;
    for(int i=0;i<n;i++)
    {
        v[i].input();
        v[i].calculate();
        if(v[i].average > highest)
```

```
        {
            highest = v[i].average;
            indexoftopper = i;
        }
    }

    cout << "Details of all students\n";
    for(int i=0;i<n;i++)
    {
        v[i].display();
    }
    cout << "Highest average: \n";
    v[indexoftopper].display();
}
```

Variations:

1.
```
void Student::grade()
{
    if(average >= 90) G='A';
    else if(average>=75) G='B';
    else if(average>50) G = 'C';
    else G = 'F';
}
```

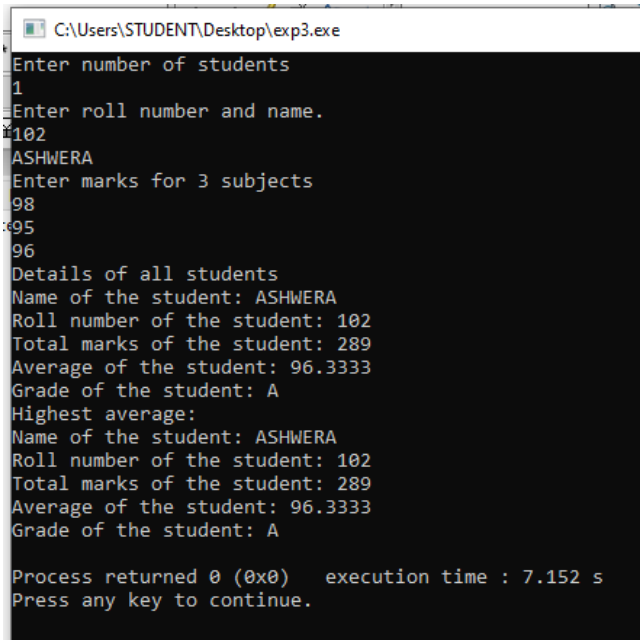
Output:

Main code:

```
C:\Users\STUDENT\Desktop\exp3.exe
Enter number of students
3
Enter roll number and name.
100
iya
Enter marks for 3 subjects
23
26
29
Enter roll number and name.
107 ashwera
Enter marks for 3 subjects
12
32
26
Enter roll number and name.
103
navya
Enter marks for 3 subjects
15
16
29
Details of all students
Name of the student: iya
Roll number of the student: 100
Total marks of the student: 78
Average of the student: 26
Name of the student: ashwera
Roll number of the student: 107
Total marks of the student: 70
Average of the student: 23.3333
Name of the student: navya
Roll number of the student: 103
Total marks of the student: 60
Average of the student: 20
Highest average:
Name of the student: iya
Roll number of the student: 100
Total marks of the student: 78
Average of the student: 26

Process returned 0 (0x0)   execution time : 25.773 s
Press any key to continue.
```

Variable:



Conclusion:

With the help of public and private access specifiers, we are able to hold better control of variables and computation on them. An array of objects of type student helps us access various attributes related to the class Student. Using the method reference operator ::, we are able to avoid passing values to public methods, and we can directly access functions corresponding to concerned objects.

Post Lab Descriptive Questions:

What is encapsulation, and how is it implemented in your program?
Encapsulation refers to the wrapping up of attributes and methods into a class together. In my program, we initialised and declared member variables and methods in the class called Student, which we later called using objects in the main() function. The class Student *encapsulated* the method and attribute in itself as a unit.

What is the difference between a constructor and a regular member function?

1. A constructor has the same name as the class. A regular member function can have any name per the convention.
2. Constructors initialize object state when an object is created. Regular member functions perform actions only.
3. Constructors do not have a return type, while functions usually return something or are void.

4. Constructors cannot be declared as static. Regular member functions can be declared as static, which makes them accessible to every entity of the class.

Why are data members declared `private` in a class?

For enforcing encapsulation and abstraction, data members are declared private in a class. Private members are only accessible from within the class itself. This ensures that the private information cannot be accessed by any member outside the class.

Date: 11/08/2025 **Signature of faculty in-charge**