Object Oriented Programming Lab

Assignment 6

Submitted by:

Navdeep Singh

2nd September 2025

Roll No: 24124073
Group: 3
Branch: Information Technology
Year: 2nd Year

Q1. Write a program with multiple classes, some of which are members of other classes, to observe the order in which constructors and destructors are called during object creation and destruction. You can use std::cout statements within the constructors and destructors to trace their execution.

Code

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

class class1 {
    public:
        class1() {
            cout << "Constructor of class1 called" << endl;
        }
        ~class1() {
            cout << "Destructor of class1 called" << endl;
        }
};

class class2 {
    public:
        class2() {
            cout << "Constructor of class2 called" << endl;
        }
        ~class2() {
            cout << "Destructor of class2 called" << endl;
        }
};

class class3 {
    class2 b;
    class1 a;
    public:
        class3() {
            cout << "Constructor of class3 called" << endl;
        }
        ~class3() {
            cout << "Destructor of class3 called" << endl;
        }
};


int main() {
    cout << "Creating object constrcurtor" << endl;
    class3 c;
    cout << "Object created." << endl;

    cout << "Exiting destrcutor called" << endl;
    return 0;
}
```

Sample Output

```
Creating object constrcurtor
Constructor of class2 called
Constructor of class1 called
Constructor of class3 called
Object created.
Exiting destrcutor called
Destructor of class3 called
Destructor of class1 called
Destructor of class2 called
```

Q2. Write a program to read and print students information using two classes and simple inheritance. Structure of the base class is given below:

Code

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

// Base class
class BasicInfo {
private:
    string s;
    int age;

protected:
    char gender;

public:
    void getBasicInfo() {
        cout << "Enter name: ";
        cin >> s;
        cout << "Enter age: ";
        cin >> age;
        cout << "Enter gender (M/F): ";
        cin >> gender;
    }

    void putBasicInfo() {
        cout << "Name: " << s << endl;
        cout << "Age: " << age << endl;
        cout << "Gender: " << gender << endl;
    }
};

// Derived class
class Student : public BasicInfo {
private:
    int totalM;
    float perc;
    char grade;

public:
    void getResultInfo() {
        cout << "Enter total marks: ";
```

```
        cin >> totalM;
        cout << "Enter percentage: ";
        cin >> perc;
        cout << "Enter grade: ";
        cin >> grade;
    }

    void putResultInfo() {
        cout << "Total Marks: " << totalM << endl;
        cout << "Percentage: " << perc << "%" << endl;
        cout << "Grade: " << grade << endl;
    }
};

int main() {
    Student st;
    cout << "--- Enter Student Information ---" << endl;
    st.getBasicInfo();
    st.getResultInfo();

    cout << "\n--- Displaying Student Information ---" << endl;
    st.putBasicInfo();
    st.putResultInfo();

    return 0;
}
```

Sample Output

```
--- Enter Student Information ---
Enter name: Navdeep
Enter age: 20
Enter gender (M/F): M
Enter total marks: 480
Enter percentage: 96
Enter grade: A

--- Displaying Student Information ---
Name: Navdeep
Age: 20
Gender: M
Total Marks: 480
Percentage: 96%
Grade: A
```

Accessibility Tables

Accessibility in Public Mode

	Private	*Protected*	*Public*
BaseClass	*NotAccessible*	*Accessible*	*Accessible*
DerivedClass	*NotAccessible*	*Accessible*	*Accessible*

Accessibility in Protected Mode

	Private	*Protected*	*Public*
BaseClass	*NotAccessible*	*Accessible*	*Accessible*
DerivedClass	*NotAccessible*	*Accessible*	*AccessibleasProtected*

Accessibility in Private Mode

	Private	*Protected*	*Public*
BaseClass	*NotAccessible*	*Accessible*	*Accessible*
DerivedClass	*NotAccessible*	*AccessibleasPrivate*	*AccessibleasPrivate*