

# Programming in Modern C++: Assignment Week 4

Total Marks : 25

Partha Pratim Das  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur, Kharagpur – 721302  
partha.p.das@gmail.com

February 14, 2025

## Question 1

Consider the following code segment.

*[MCQ, Marks 2]*

```
#include<iostream>
using namespace std;
class myClass{
    static int s;
public:
    myClass(int _s) : s(_s) { }
    void incr(){ s = s+10; }
    void print(){ cout << s; }
};
int myClass::s = 10;
int main(){
    myClass o1(5), o2(10);
    o1.incr();
    o2.incr();
    o2.print();
    return 0;
}
```

What will be the output/error?

- a) 15
- b) 20
- c) 30
- d) Compilation Error: myClass::s is a static data member; it can only be initialized at its definition

**Answer:** d)

**Explanation:**

Static member cannot be initialized from the class constructor. Hence, it will give compilation error.

## Question 2

Consider the following code segment.

[MCQ, Marks 2]

```
#include<iostream>
using namespace std;
class Point{
    int x, y;
public:
    Point(int r=0, int i=0) : x(r), y(i) {}
    Point& operator<< (const Point& c){ //Line-1
        cout << x+c.x << "," << y+c.y << endl;
        return *this;
    }
    friend Point& operator<<(ostream& os, Point& c);
};
Point& operator<<(ostream& os, Point& c){ //Line-2
    cout << c.x << "," << c.y << endl;
    return c;
}
int main(){
    Point c1(2,5), c2(4,6);
    cout << c1 << c2;
    return 0;
}
```

What will be the output?

- a) 2,5  
4,6
- b) 6,5  
2,11
- c) 6,11  
2,5
- d) 2,5  
6,11

**Answer:** d)

**Explanation:**

The evaluation of `cout` statement is done from left to right. So, the operator overloading function at **Line-2** will be called first, which will print 2,5 and return the object `c1`. Then the operator function of **Line-1** will be called, which will print 6,11.

### Question 3

Consider the following code segment.

*[MCQ, Marks 2]*

```
#include <iostream>
using namespace std;
int var = 0;
namespace name {
    int var = 2;
}
int main() {
    int var = 1;
    {
        using namespace name;
        cout << ::var << " " << var << " " << name::var; // LINE-1
    }
    return 0;
}
```

What will be the output/error?

- a) 0 1 2
- b) 0 2 2
- c) 2 0 1
- d) Compilation Error: reference to 'var' is ambiguous

**Answer:** a)

**Explanation:**

When there are multiple instances of the same variable, the local instance will get higher priority. So, var will be printed as 1. To access global variable, we use `::var`. For the namespace variable, it is qualified by the namespace `name`. So, cout statement at LINE-1 will print 0 1 2.

## Question 4

Consider the following code segment.

[MCQ, Marks 2]

```
#include <iostream>
using namespace std;
class myClass {
    static int X;
    static int Y;
public:
    ----- void print() {           //LINE-1
        cout << X << " " << Y;
    }
    void setX(int a){
        X=a;
    }
    void setY(int a){
        Y=a;
    }
};
int myClass::X = 10;
int myClass::Y = 10;
int main() {
    myClass t1, t2;
    t1.setX(4);
    t2.setY(5);
    myClass::print();
    return 0;
}
```

Fill in the blank at LINE-1 such that the output will be 4 5.

- a) mutable
- b) static
- c) const
- d) friend

**Answer:** b)

**Explanation:**

A function can be called using the class name only when it is **static** function. So, the function **print** should be declared as static.

## Question 5

Consider the following code segment.

*[MSQ, Marks 2]*

```
#include<iostream>
using namespace std;
int x=10;
namespace e{
    namespace e1{
        int x=15;
    }
    int x=5;
}
int main(){
    -----; //LINE-1
    cout << x;
    return 0;
}
```

Fill in the blank at LINE-1 such that the program will print 15.

- a) `using e::e1::x`
- b) `using e::e1`
- c) `using namespace e::e1`
- d) `using namespace e`

**Answer:** a)

**Explanation:**

The variable `x` declared in namespace `e1` needs to be made available in order to print 15 as output. This can be done by filling up in LINE-1 as `using e::e1::x`.

*This question is intentionally made as MSQ*

## Question 6

Consider the following code segment.

[MSQ, Marks 2]

```
#include<iostream>
using namespace std;
class constTest{
    ----- x; //LINE-1
public:
    constTest(int _x) : x(_x) {}
    void set(int a) const{
        x = a;
    }
    void print() const{
        cout << x << endl;
    }
};
int main(){
    const constTest m(5);
    m.set(10);
    m.print();
    return 0;
}
```

Fill in the blank at LINE-1 such that the output is 10.

- a) int mutable
- b) mutable int
- c) volatile int
- d) const int

**Answer:** a), b)

**Explanation:**

To change the value of the data member of a constant object, we need to declare the data member as mutable. So, the syntaxes are `mutable int` or `int mutable`.

## Question 7

Consider the following code segment.

*[MCQ, Marks 2]*

```
#include<iostream>
using namespace std;
namespace name{
    class Student{
        int roll;
    public:
        Student(int x) : roll(x) {}
        void print(){ cout << roll; }
    };
}
int main(){
    -----; //LINE-1
    s.print();
    return 0;
}
```

Fill in the blank at LINE-1 such that the output is 5.

- a) `name::Student s(5)`
- b) `Student s(5)`
- c) `name.Student s(5)`
- d) `using name::Student s(5)`

**Answer:** a)

**Explanation:**

The class is declared under namespace `name`. So, correct declaration of an object of class `Student` will be `name::Student s(5)`.

## Question 8

Consider the following code segment.

[MSQ, Marks 2]

```
#include<iostream>
using namespace std;
class Point{
    int x, y;
    public:
        Point(int _x, int _y) : x(_x), y(_y) {}
        _____; //LINE-1
};
Point& operator<< (ostream& os, Point& p){
    cout << "(" << p.x << "," << p.y << ")" << endl;
    return p;
}
int main(){
    Point pt(2,5);
    cout << pt;
    return 0;
}
```

Fill in the blank at LINE-1 such that the program will print (2,5).

- a) Point& operator<< (ostream&, Point&)
- b) friend Point& operator<< (ostream&, Point&)
- c) Point& friend operator<< (ostream&, Point&)
- d) const Point& operator<< (ostream&, Point&)

**Answer:** b)

**Explanation:**

The operator is overloaded for the class Point. So, the overloaded function should be a friend of class Point. The correct syntax to declare the function as friend is `friend Point& operator<< (ostream&, Point&)`.

This question is intentionally made as MSQ



## Question 9

Consider the following code segment.

[MCQ, Marks 2]

```
#include<iostream>
using namespace std;
class A{
    static int a;
    public:
        int get(){ return a; }
        -----; //LINE-1
};
int A::a = 0;
class B{
    int b;
    public:
        B(int y) : b(y) {}
        void print(){
            A::a = 10;
            cout << A::a-b << " " << A::a+b;
        }
};
int main(){
    B t2(5);
    t2.print();
    return 0;
}
```

Fill in the blank at LINE-1 such that the program will print 5 15.

- a) static class B
- b) friend class B
- c) class friend B
- d) using class B

**Answer:** b)

**Explanation:**

Here, class B is accessing private static data member of class A. This can only be possible if class B is a friend of class A or B::print() function is a friend of class A. But we can't declare B::print() as friend at LINE-1 because there is no forward declaration of class B. Hence, correct option is b.

# Programming Questions

## Question 1

Consider the program below which defines a class `Database`. Complete the program with the following instructions.

- Fill in the blank at LINE-1 to complete the declaration of member variable `ins`.
- Fill in the blank at LINE-2 to specify the return type of `createIns()` function.
- Fill in the blank at LINE-3 to call the `createIns()` function with parameter `i`.

The program must satisfy the given test cases.

*Marks: 3*

```
#include<iostream>
using namespace std;
class Database{
    int id;
    ----- Database *ins;                //LINE-1
    Database(int i) : id(i) {}
public:
    int getIns(){ return id; }
    ----- createIns(int i){            //LINE-2
        if(!ins)
            ins = new Database(i);
        return ins;
    }
    ~Database();
};
Database *Database::ins = 0;
void fun(int i){
    Database *s = -----;                //LINE-3
    cout << s->getIns();
}
Database::~~Database(){ cout << id; }
int main(){
    int a,b;
    cin >> a >> b;
    Database *s = Database::createIns(a);
    cout << s->getIns();
    fun(b);
    return 0;
}
```

### Public 1

Input: 4 5

Output: 44

### Public 2

Input: 5 10

Output: 55

## Private 1

Input: 14 20

Output: 1414

### Answer:

LINE-1: `static`

LINE-2: `static Database*`

LINE-3: `Database::createIns(i)`

### Explanation:

This is an example of singleton class. So, the object of `class Singleton` will be instantiated only once and that is used throughout the program. So, the variable at **LINE-1** should be declared as `static`. The `createIns()` function is returning the pointer data member of class `Database`. Hence, **LINE-2** will be filled as `static Database*`. The `createIns()` function can be called at **LINE-3** as `Database::createIns(i)`.

## Question 2

Consider the following program.

- Fill in the blanks at LINE-1 to complete forward declaration.
- Fill in the blank at LINE-2 with appropriate function declaration so that function `calculate` can access private data member of `TotalAmount` class.

The program must satisfy the sample input and output.

*Marks: 3*

```
#include<iostream>
using namespace std;
----- //LINE-1
class TotalAmount{
    double prinAmt;
    double amt = 0;
public:
    TotalAmount(double p) : prinAmt(p){}
    double calculate(Interest&);
};
class Interest{
    double in;
public:
    Interest(double i) : in(i){ }
    ----- //LINE-2
};
double TotalAmount::calculate(Interest &i){
    amt = prinAmt * (1 + i.in / 100);
    return amt;
}
int main(){
    double i, j;
    cin >> i >> j;
    TotalAmount m(i);
    Interest in(j);
    cout << "Matured Amount: " << m.calculate(in);
    return 0;
}
```

### Public 1

Input: 1000 5

Output: Matured Amount: 1050

### Public 2

Input: 5000 8

Output: Matured Amount: 5400

### Private

Input: 10000 6

Output: Matured Amount: 10600

**Answer:**

LINE-1: `class Interest;`

LINE-2: `friend double TotalAmount::calculate(Interest&);`

OR

LINE-2: `friend class TotalAmount;`

**Explanation:**

As we can see, the function `calculate()` is taking parameter of type class `Interest` which is not yet declared, forward declaration for class `Interest` is needed. The function `calculate()` is accessing private data member of class `TotalAmount`. It can be done only if the function or the class `TotalAmount` is friend of class `Interest`.

### Question 3

Consider the following program. Fill in the blanks as per the instructions given below:

- at LINE-1 to complete operator overload function,
- at LINE-2 and LINE-3 to calculate subtraction of two `position` class.

such that it will satisfy the given test cases.

*Marks: 3*

```
#include<iostream>
using namespace std;
class position{
    int x, y;
public:
    position(int a, int b) : x(a), y(b) {}
    -----(const position& p1){ //LINE-1
        position p(0,0);
        p.x = -----; //LINE-2
        p.y = -----; //LINE-3
        return p;
    }
    void print(){ cout << "(" << x << ", " << y << ")"; }
};
int main(){
    int x1,y1,x2,y2;
    cin >> x1 >> y1 >> x2 >> y2;
    position p1(x1,y1), p2(x2,y2), p3(0,0);
    p3 = p1-p2;
    p3.print();
    return 0;
}
```

#### Public 1

Input: 5 8 6 7  
Output: (-1, 1)

#### Public 2

Input: 6 7 2 5  
Output: (4, 2)

#### Private

Input: 5 2 6 1  
Output: (-1, 1)

**Answer:**

LINE-1: position operator-

LINE-2: x - p1.x OR this->x - p1.x

LINE-3: y - p1.y OR this->y - p1.y

**Explanation:**

The addition operator needs to be overloaded and need to return the resultant position object. So, overloaded operator function header will be filled as `position operator-(const position& p1)`. The calculation at LINE-2 and LINE-3 will be filled as,

Line-2: x - p1.x or this->x - p1.x

Line-3: y - p1.y or this->y - p1.y