Programming in Modern C++: Assignment Week 4

Total Marks : 20

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

August 10, 2023

Question 1

Consider the following program. *[MSQ, Marks 2]*

```
#include<iostream>
using namespace std;
class Student{
    string name = "Raj";
    public:
        ______________________________; //Line-1
};
void show(const Student &t){
    cout << "Hello " << t.name ;
}
int main(){
    Student t;
    show(t);
    return 0;
}
```

Fill in the blank at `LINE-1` such that the program will print ”Hello Raj”.

a) `void show(const Student&)`

b) `friend void show(const Student&)`

c) `static void show(const Student&)`

d) `void friend show(const Student&)`

Answer: b), d)
Explanation:
The global function `show()` is accessing a private member of the class `Student`. So, the function has to be friend of class `Student`. So, correct options are b) and d).

Question 2

Consider the code segment given below. *[MCQ, Marks 2]*

```
#include<iostream>
using namespace std;
class statC{
    static double d=9.81;
    public:
        void display() { cout << d << endl; }
};
int main(){
    statC s;
    s.display();
    return 0;
}
```

What will be the output/error?

a) `9.81`

b) `9`

c) `0`

d) `Compilation error:  C++ forbids in-class initialization of non-const static member`

Answer: d)
Explanation:
In-class initialization is not allowed for non-const static members. So it will give a compilation error.

Question 3

Consider the following code segment. *[MCQ, Marks 2]*

```
#include<iostream>
using namespace std;
class Test1{
    int id;
    public:
        Test1(int x) : id(x) { cout << id << " "; }
        ~Test1() { cout << id << " "; }
        void fun(){
            static Test1 t2(5);
        }
};
int main(){
    Test1 t1(1);
    t1.fun();
    return 0;
}
```

What will be the output?

a) `1 5 5 1`

b) `1 5 1 5`

c) `5 1 1 5`

d) `5 1 5 1`

Answer: b)
Explanation:
The lifetime of a static object will end only after the whole program execution. So, the constructor of object `t1` from the main function will be called first, which will print 1. After that, constructor from function `fun()` will be called, which will print 5. Then main function object will be destroyed. At last, the static object will be destroyed.

Question 4

Consider the code segment given below. *[MCQ, Marks 2]*

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

What will be the output?

a) `0 1 2`

b) `1 0 2`

c) `0 2 1`

d) `2 1 0`

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

Question 5

Consider the code segment. *[MSQ, Marks 2]*

```
#include <iostream>
using namespace std;
class Test {
    static int X;
public:
    static void print() {
        cout << X;
    }
    _______________ update(int a){ //Line-1
        X=a;
    }
};
int Test::X = 10;
int main() {
    Test::update(4);
    Test::print();
    return 0;
}
```

Fill in the blank at `LINE-1` such that the program will print `4`.

a) `void static`

b) `void const`

c) `static void`

d) `void`

Answer: a), c)
Explanation:
A function can be called using the class name only when it is `static` function. So, the function `update` should be declared as `static`. Hence, correct options are a) and c).

Question 6

Consider the code segment given below. *[MCQ, Marks 2]*

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

Fill in the blank at `LINE-1` so that the program will print 5.

a) `using namespace ns::x`

b) `using ns::x`

c) `using namespace ns`

d) `using namespace ::x`

Answer: b)
Explanation:
The namespace variable `x` needs to be made available in order to print 5 as output. This can be done by filling up in `LINE-1` as `using ns::x`.

Question 7

Consider the code segment below. *[MSQ, Marks 2]*

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

Fill in the blank at `LINE-1` so that the program will print `0`.

a) `mutable int`

b) `int mutable`

c) `const int`

d) `int`

Answer: a), b)
Explanation: To change the value of a data member of a constant object, we need to declare the data member as `mutable`. So, the syntax is `mutable int` or `int mutable`.

Question 8

Consider the code segment given below. *[MCQ, Marks 2]*

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

Fill in the blank at `LINE-1` so that the program will print `5`.

a) `name::myClass m(5)`

b) `myClass m(5)`

c) `using name::myClass m(5)`

d) `name.myClass m(5)`

Answer: a)
Explanation: The class is declared under namespace name. So, correct declaration of an object of class `myClass` will be `name::myClass m(5)`.

Question 9

Consider the code segment given below. *[MSQ, Marks 2]*

```
#include<iostream>
using namespace std;
class classA{
    static int a;
    public:
        int get(){ return a; }
        _____________________; //LINE-1
};
int classA::a = 0;
class classB{
    int b;
    public:
        classB(int y) : b(y) {}
        void print(){
            classA::a = 4;
            cout << b << " " << classA::a;
        }
};
int main(){
    classB t2(5);
    t2.print();
    return 0;
}
```

Fill in the blank at `LINE-1` so that the program will print `5 4`.

a) `friend class classB`

b) `using class classB`

c) `friend void classB::print()`

d) `using void classB::print()`

Answer: a)
Explanation:
Here, class `classB` is accessing a private static data member of class `classA`. This can only be possible if class `classB` is a friend of class `classA` or `classB::print()` function is a friend of class `classA`. But we can't declare `classB::print()` as friend at `LINE-1` because there is no forward declaration of class `classB`. Hence, the correct option is a).
Intentionally made as MSQ

Programming Questions

Question 1

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

- Complete the variable declaration at `LINE-1`,
- Complete the function prototype at `LINE-2` and `LINE-3` with appropriate keywords

such that it will satisfy the given test cases. *Marks: 3*

```
#include<iostream>
using namespace std;
class Employee{
    const int id;
    string name;
    ________ int salary; //LINE-1
    public:
        Employee(int a, string b, int c) : id(a), name(b), salary(c) {}
        void updateSal(int x) ______{ salary += x; } //LINE-2
        void print() ______{ cout << id << " : " << name << " : " << salary; } //LINE-3
};
int main(){
    string n;
    int i, m, u;
    cin >> i >> n >> m >> u;
    const Employee e1(i, n, m);
    e1.updateSal(u);
    e1.print();
    return 0;
}
```

Public 1

```
Input: 1 Raj 10000 1000
Output: 1 : Raj : 11000
```

Public 2

```
Input: 2 Zakir 50000 5000
Output: 2 : Zakir : 55000
```

Private

```
Input: 3 Sam 1000 300
Output: 3 : Sam : 1300
```

Answer:

```
LINE-1:  mutable
LINE-2:  const
LINE-3:  const
```

Explanation:

The object of class `Employee` is declared as constant. So, all functions of class `Employee` should be declared as `const` in order to access them using the constant object. So, `LINE-2` and `LINE-3` will be filled as `const`. The data member `salary` of constant object is being changed using function `updateSal()`. Hence, `LINE-1` should be filled as `mutable`.

Question 2

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

- at LINE-1 with appropriate forward declaration,
- at LINE-2 with appropriate statement

such that it will satisfy the given test cases. *Marks: 3*

```
#include<iostream>
using namespace std;
___________; //LINE-1
class A{
    int a_ = 0;
    public:
        A(int x) : a_(x) {}
        int mulB (B&);
        int subtractB (B&);
};
class B{
    int b_;
    public:
        B(int y) : b_(y) { }
        ________________; //LINE-2
};
int A::mulB(B &b) {
    return (a_ * b.b_);
}
int A::subtractB(B &b) {
    return (a_ - b.b_);
}
int main(){
    int x, y;
    cin >> x >> y;
    A t1(x);
    B t2(y);
    cout << t1.mulB(t2) << " " << t1.subtractB(t2);
    return 0;
}
```

Public 1

```
Input: 3 4
Output: 12 -1
```

Public 2

```
Input: 2 7
Output: 14 -5
```

Private

```
Input: 8 4
Output: 32 4
```

Answer:
`LINE-1: class B`
`LINE-2: friend class A`
Explanation:
`LINE-1` should be filled with forward declaration of `class B` because the `class B` is used in `class A`. As, both functions of `class A` are accessing private member of `class B`, `class A` should be a friend of `class B`. So, `LINE-2` should be filled as `friend class A`.

Question 3

Consider the following program. Fill in the blanks at `LINE-1`, `LINE-2` and `LINE-3` with appropriate statements such that it will satisfy the given test cases. *Marks: 3*

```
#include<iostream>
using namespace std;
class Singleton{
    int data;
    __________________ *ins;                  //LINE-1
    Singleton(int i) : data(i) {}
    public:
        int get(){ return data; }
        _________________ createIns(int i){      //LINE-2
            if(!ins)
                ins = ___________________;    //LINE-3
            return ins;
        }
        ~Singleton(){ cout << data; }
};
Singleton *Singleton::ins = 0;
void fun(int x){
    Singleton *s = Singleton::createIns(x+5);
    cout << s->get();
}
int main(){
    int i, j;
    cin >> i >> j;
    Singleton *s = Singleton::createIns(i);
    cout << s->get();
    fun(j);
    return 0;
}
```

Public 1

```
Input: 1 2
Output: 11
```

Public 2

```
Input: 2 3
Output: 22
```

Private

```
Input: 3 5
Output: 33
```

Answer:

```
LINE-1:  static Singleton
LINE-2:  static Singleton*
LINE-3:  new Singleton(i)
```

Explanation:
The pointer variable `ins` needs to be declared as `static` so that the same instance can be used by all objects. The function `createIns()` is returning `ins` variable which is of type `static Singleton*`. Hence, `LINE-2` should be filled as `static Singleton*`. We need to create an object of type class `Singleton` which can be done at `LINE-3` as `new Singleton(i)`;