Programming in 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 25, 2020

Question 1

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

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

class myClass {
    int data;
public:
    myClass(int x) : data(x) {}
    ________________________;    // LINE-1
};

void display(const myClass &m) {
    cout << m.data << endl;
}

int main() {
    myClass m(10);

    display(m);

    return 0;
}
```

This program will give error without `LINE-1`. Fill in the blank at `LINE-1` to avoid any compilation error.

a) `friend void display(const myClass&)`

b) `void friend display(const myClass&)`

c) `void display(const myClass&)`

d) `friend display(const myClass&)`

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

Question 2

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

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

class A {
    int data;
public:
    A(int x) : data(x) { cout << data << " "; }
    ~A() { cout << data << " "; }
    void show() {
        static A a(5);
    }
};

int main() {
    A a1(10);

    a1.show();

    return 0;
}
```

What will be the output of the following code?

a) 5 10 5 10

b) 10 5 10 5

c) 10 5 5 10

d) 5 10 10 5

Answer: b)
Explanation:
The lifetime of a local `static` object in a function is constructed when the function is first called and will end only after the whole program execution. So, constructor of object `a1` from the main function will be called first which will print 10. After that constructor from function `show()` will be called which will print 5. Then main function object will be destroyed. At last, static object will be destroyed.

Question 3

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

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

class Complex {
    int re, im;
public:
    Complex(int r, int i) : re(r), im(i) { }
    Complex& operator++() {   // LINE-1
        ++re;
        return *this;
    }
    Complex operator++(int) { // LINE-2
        Complex c(re, im);
        ++im;
        return c;
    }
    void display() { cout << re << " " << im << endl; }
};

int main() {
    Complex c(5, 5);

    ++c;
    Complex c1 = c++;

    c1.display();

    return 0;
}
```

What will be the output?

a) 5 5

b) 6 6

c) 6 5

d) 5 6

Answer: c)
Explanation:
Pre-increment operator is applied to `Complex` object `c` which will call operator function defined in `LINE-1` which will increment `re` value by 1. Then post-increment operator is applied which will increment `im` value by 1 but after assignment is done to `c1`. So, it will print as `6 5`.

Question 4

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

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

class myClass {
    static int i = 5;
public:
    void display() { cout << i << endl; }
};

int main() {
    myClass m;

    m.display();

    return 0;
}
```

What will be the output/error?

a) `5`

b) `0`

c) `<Unpredicted value>`

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

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

Question 5

What will be the output of the following program. *[MCQ, Marks 2]*

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

class Complex {
    int re, im;
public:
    Complex(int r = 0, int i = 0) : re(r), im(i) { }
    Complex& operator<< (const Complex& c) {          // LINE-1
        cout << re + c.re << " " << im + c.im << endl;
        return *this;
    }
    friend Complex& operator<<(ostream& os, Complex& c);
};

Complex& operator<<(ostream&, Complex& c) {        // LINE-2
    cout << c.re << " " << c.im << endl;
    return c;
}

int main() {
    Complex c1(2, 5), c2(4, 6);

    cout << c1 << c2;

    return 0;
}
```

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, operator function at `LINE-2` will be called first which will print `2 5` and return the object `c1`. Then operator function of `LINE-1` will be called which will print `6 11`.

Question 6

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

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

int var = 0;
namespace name {
    int var = 2;
}

int main() {
    using namespace name;
    int var = 1;

    cout << ::var << " " << var << " " << name::var; // LINE-1

    return 0;
}
```

What will be the output?

a) `0 1 2`

b) `1 0 2`

c) `0 2 1`

d) `1 2 0`

Answer: a)
Explanation:
When there are multiple instances of the same variable, the local instance will get higher priority. So, `var` is the local variable which 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 be printed as `0 1 2`.

Question 7

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

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

class Test {
    static int X;
public:
    static void print() {
        cout << X;
    }
    static update(int a) {  // LINE-1
        X = a;
    }
};

int Test::X = 10;

int main() {
    Test::update(4);
    Test::print();

    return 0;
}
```

Identify the correct replacement/s of `LINE-1` for output `4`.

a) `void static update(int a)`

b) `static void update(int a)`

c) `void update(int a)`

d) `friend void update(int a)`

Answer: a), b)
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`. So, answers are a), b).

Question 8

Consider the program below. *[MCQ, Marks 2]*

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

class myClass {
    int X;
    static myClass *instance;
    myClass(int i) : X(i) { }
public:
    int getVal() { return X; }
    static myClass * createInstance(int x) {
        if (!instance) {
            instance = new myClass(x);
        }
        return instance;
    }
};

myClass *myClass::instance = 0;

void foo() {
    myClass *s = myClass::createInstance(1);
    cout << s->getVal() << " ";
}

void fun() {
    myClass *s = myClass::createInstance(2);
    cout << s->getVal() << " ";
}

int main() {
    foo();
    fun();

    myClass *s = myClass::createInstance(3);
    cout << s->getVal() << " ";
    return 0;
}
```

What will be the output?

a) `1 2 3`

b) `3 2 1`

c) `1 1 1`

d) `3 3 3`

Answer: c)
Explanation:
This is the typical example of Singleton class. Instance of class `myClass` is created when it is called for the first time. Next onwards, same instance is returned every time the `createInstance` function is called. So, output will be `1 1 1`.

Question 9

Consider the program below. *[MCQ, Marks 2]*

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

int x = 10;
namespace e {
    int x = 5;
}

int main() {
    ____________________     // LINE-1
        cout << x;

    return 0;
}
```

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

a) `using namespace e;`

b) `using namespace e::x;`

c) `using e::x;`

d) `using namespace ::x;`

Answer: c)
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 e::x`.

Programming Questions

Question 1

Consider the following program. Fill in the blanks in `LINE-1` for forward declaration and in `LINE-2` to make available private variable of class `B` to the member functions of class `A` such that it will satisfy sample input and output. *Marks: 3*

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

_____________;              // LINE-1

class A {
    int a_ = 0;
public:
    A(int x) : a_(x) { }
    int addB(B&);
    int subtractB(B&);
};

class B {
    int b_ = 5;
public:
    _____________;          // LINE-2
};

int A::addB(B &b) {
    return (a_ + b.b_);
}

int A::subtractB(B &b) {
    return (a_ - b.b_);
}

int main() {
    int x;
    cin >> x;

    A t1(x);
    B t2;

    cout << t1.addB(t2) << " " << t1.subtractB(t2);

    return 0;
}
```

Public 1

```
Input: 10
Output: 15 5
```

Public 2

```
Input: 5
Output: 10 0
```

Private

```
Input: 7
Output: 12 2
```

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 2

Consider the following program and fill in the blanks in `LINE-1` for overloading of multiplication operator, in `LINE-2` and `LINE-3` to calculate multiplication of two complex numbers. Consider sample input and output. *Marks: 3*

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

class Complex {
    int re, im;
public:
    Complex(int r = 0, int i = 0) : re(r), im(i) { }
    ____________________(const Complex &c) { // LINE-1
        int x, y;
        x = ____________________;            // LINE-2
        y = ____________________;            // LINE-3

        Complex t1(x, y);
        return t1;
    }

    void show() {
        cout << re << " " << im;
    }
};

int main() {
    int x1, x2, y1, y2;
    cin >> x1 >> y1 >> x2 >> y2;

    Complex c1(x1, y1), c2(x2, y2);
    Complex c3 = c1 * c2;

    c3.show();

    return 0;
}
```

Public 1

```
Input: 1 1 2 2
Output: 0 4
```

Public 2

```
Input: 1 2 3 4
Output: -5 10
Hello Student
```

Private

```
Input: 1 2 1 2
Output: -3 4
```

Answer:
`LINE-1`: `Complex operator*`
`LINE-2`: `re * c.re - im * c.im`
`LINE-3`: `re * c.im + im * c.re`

Explanation:
We have to overload multiplication operator and return a `Complex` class variable. So, `LINE-1` can be filled with `Complex operator*`. `LINE-2` and `LINE-3` are filled with multiplication calculation of two complex number. So, it can be filled as,
`LINE-2`: `re * c.re - im * c.im`
`LINE-3`: `re * c.im + im * c.re`

Question 3

Consider the following program and fill in the banks in `LINE-1` with appropriate overloaded operation and `LINE-2` with appropriate parameter for `fun()` function such that it matches the given test cases. *Marks: 3*

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

class myClass {
public:
    int data;
    myClass(int x) : data(x) { }
    myClass operator==(myClass &m1) {
        myClass m(0);
        ____________________;    // LINE-1
            return m;
    }
};

void fun(_____________________) { // LINE-2
    cout << m.data << endl;
}

int main() {
    int i, j, k;
    cin >> i >> j >> k;

    myClass m1(i), m2(j), m3(k);
    fun(m1 == m2 == m3);

    return 0;
}
```

Public 1

```
Input: 0 2 2
Output: 0
```

Public 2

```
Input: 8 8 1
Output: 1
```

Private

```
Input: 5 0 5
Output: 0
```

Answer:

`LINE-1`: `m.data = (m1.data == data ?  1 :  0);` OR
`m.data = (m1.data == this->data ?  1 :  0);`

`LINE-2`: `const myClass &m` OR `myClass &m`

Explanation:
In `m1 == m2 == m3`, the result of `m1 == m2` is compared with `m3`. From the test cases we find that result of this comparison is `1` if `m3` is `1`, otherwise it is `0`. So the outcome of the comparison must give `1` on equality. Hence, `LINE-1` will be filled with ternary operator which will compare the value of both object passed in the overloaded operator and return `1` if it matches else returns `0`. `LINE-2` should be filled with `const` parameter as the `fun()` need not change the value of `data`. However, using `const` is not mandatory.

Question 4

Consider the below program. Fill in the blanks at `LINE-1, LINE-2 and LINE-3` by following given instructions such that it satisfies the given test cases. *Marks: 3*

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

class myClass {
    int data;
    __________ myClass *t;        // LINE-1 Complete the declaration
    myClass(int x) : data(x) { }
public:
    __________ create(int x) {    // LINE-2 Mention return type of the function
        if (!t)
            t = __________;       // LINE-3 Allocate memory towards object t
            return t;
    }
    void show() {
        cout << data;
    }
};

myClass *myClass::t = 0;

int main() {
    int x, y;
    myClass *m1, *m2;

    cin >> x >> y;
    m1 = myClass::create(x);
    m2 = myClass::create(y);

    m1->show();
    m2->show();

    return 0;
}
```

Public 1

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

Public 2

```
Input: 2 5
Output: 22
```

Private

```
Input: 1 5
Output: 11
```

Answer:

```
LINE-1:  static
LINE-2:  static myClass*
LINE-3:  new myClass(x)
```

Explanation:
The variable `t` should be declared as `static` so that the first instance of `myClass` will always present throughout the program. The function is returning the member `t`. So `LINE-2` is filled as `static myClass*` . Memory allocation in `LINE-3` should be done as `t = new myClass(x);`.