Programming in Modern C++: Assignment Week 2

Total Marks : 25

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

January 12, 2023

Question 1

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

```
#include <iostream>
using namespace std;
char add(char c1 = 'a') { return c1; }
char add(char c1 = 'a', char c2 = 'b') { return c1 + c2 - 'a';}
char add(char c1 = 'a', int d1 = 100){ return c1 + d1 - 'a'; }
char add(char c1 = 'a', char c2 = 'b', char c3) { return c1 + c2 + c3 - 'a'; }
int main() {
    char c = add('o', 'k');
    cout << c << endl;
    return 0;
}
```

What will be the output/error(s)?

a) `y`

b) `z`

c) `Compilation Error:  default argument missing for "char add(char, char, char)"`

d) `Compilation Error:  call of overload "add(char, char)" is ambiguous`

Answer: c), d)
Explanation:
The call `add('o', 'k')`, can invoke both the following overloads of `add` function:
`char add(char c1 = 'a', char c2 = 'b') { ... }`
`int add(char c1 = 'a', char c2 = 'b', char c3) { ... }`
Thus, the call is ambiguous. Also, the second function definition itself has an error due to the missing default argument for the third parameter. Hence, correct options are c and d.

Question 2

Consider the following code segment. *[MCQ, Mark 1]*

```
#include <iostream>
using namespace std;
#define SQR(x) (x)*(x)

int main() {
    int a=3;
    cout << SQR(a++) << endl;
    return 0;
}
```

What will be the output?

a) `12`

b) `25`

c) `9`

d) `16`

Answer: a)
Explanation:
In a macro call, the arguments get substituted blindly, and then evaluated. So, the evaluation of the cout expression will be done as follows:

```
cout << SQR(a++) << endl;
cout << (a++)*(a++) << endl;
cout << (4)*(3) << endl;
cout << 12 << endl;
```

Hence, correct option is a.

Question 3

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

```
#include<iostream>
#define X 1
using namespace std;
int main(){
    int i;
    const int i1 = 2;
    const int i2 = i1; //LINE-1
    i2 = X;            //LINE-2
    i = i1;            //LINE-3
    i1 = i;            //LINE-4
    return 0;
}
```

Which line/s will give you an error?

a) `LINE-1`

b) `LINE-2`

c) `LINE-3`

d) `LINE-4`

Answer: b), d)
Explanation:
const data can only be initialized/updated at the time of declaration. Hence, `LINE-2` and `LINE-4` gives a compilation error.

Question 4

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

```
#include<iostream>
using namespace std;
int main(){
    int a = 5;
    int &b = a+1;
    a = a*b;
    cout << a << " " << b;
    return 0;
}
```

What will be the output/error?

a) `36`

b) `30`

c) `25`

d) `Compilation Error:  invalid initialization of non-const reference`

Answer: d)
Explanation:
A reference variable can be initialized with an expression if it is a constant reference. Hence, it will give a compilation error.

Question 5

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

```
#include <iostream>
using namespace std;
int& func(int& i) {     //LINE-1
    return i = i+5;
}
int main() {
    int x = 1, y = 2;
    int& z = func(x);
    cout << x << " " << z << " ";
    func(x) = y;
    cout << x << " " << z;
    return 0;
}
```

What will be the output?

a) `6 6 2 2`

b) `6 6 7 7`

c) `1 1 2 2`

d) `1 1 7 7`

Answer: a)
Explanation:
The increment of the formal parameter `i` is reflected on the actual variable `x` because it is passed as a reference. However, not as a constant reference, since the formal parameter is modified within the function. So, first cout statement will print `6 6`.
The statement `int& z = func(x);` requires the return type to be a reference type. This statement will modify the value of x and z with the value of y. Hence, a) is the correct option.

Question 6

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

```
#include <iostream>
using namespace std;
void compute(int n1, int n2, ________, ________){ //LINE-1
    n3 = n1 + n2;
    *n4 = n1 * n2;
}
int main(){
    int a = 100, b = 200, c = 0, d = 0;
    compute(a, b, c, &d); //LINE-2
    cout << c << ", ";
    cout << d;
    return 0;
}
```

Choose the appropriate option to fill in the blanks at `LINE-1`, such that the output of the code would be: `300 20000`.

a) `int n3, int* n4`

b) `int& n3, int *n4`

c) `int* n3, int* n4`

d) `int& n3, int& n4`

Answer: b)
Explanation:
Since the changes made in `n3, *n4` in function `compute()` need to be reflected in the variables `c, d` in `main()`, these are either pass-by-reference or pass-by-address.
From call at `LINE-1`, it can be observed that `c` is passed-by-reference and `d` is passed-by-address. Thus, the header of compute() must be:
void compute(int n1, int n2, int& n3, int* n4)

Question 7

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

```
#include <iostream>
using namespace std;
int main() {
    int a = 2, *b;
    *b = 5;
    int * const ptr;    // LINE-1
    ptr = b;            // LINE-2
    cout << *ptr;
    return 0;
}
```

What will be the output/error?

a) `<garbage value>`

b) `5`

c) `Compilation Error at LINE-1: uninitialized const 'ptr'`

d) `Compilation Error at LINE-2: assignment of read-only variable 'ptr'`

Answer: c), d)
Explanation:
Since the pointer `ptr` is a constant, we have to initialize the pointer while declaring. So, we will get a compilation error at `LINE-1`.
As `ptr` is a constant pointer, the it cannot be initialized after the declaration. Therefore, it also generates a compilation error at `LINE-2`.

Question 8

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

```
#include <iostream>
using namespace std;
void fun(int a = 5) { cout << a << endl; }              //LINE-1
int fun(int x = 10) { cout << x << endl; return 0; }    //LINE-2
int main() {
    fun();
    return 0;
}
```

What will be the output/error?

a) `5`

b) `10`

c) `5`
`10`

d) `Compilation error at LINE-2:  ambiguating new declaration of 'int fun(int)'`

Answer: d)
Explanation:
Since the return type of a function has no effect on function overloading, the definition at of function `fun` at `LINE-2` is ambiguous.

Question 9

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

```
#include<iostream>
using namespace std;
struct complex{
    int re, im;
    void show(){ cout << re << " + i" << im; }
};
_____________________________________{ //Line-1
    c2.re = c1.re+c2.re;
    c2.im = c1.im+c2.im;
    return c2;
}
int main(){
    struct complex c1={2,5},c2{3,-2};
    struct complex t = c1 + c2;
    t.show();
    return 0;
}
```

Fill in the blank at `LINE-1` such that the program will print `5 + i3`

a) `complex operator+(complex &c1, complex &c2)`

b) `complex operator+(const complex &c1, const complex &c2)`

c) `operator+(complex &c1, complex &c2)`

d) `complex +(complex &c1, complex &c2)`

Answer: a)
Explanation:
We need to overload the addition operator for the structure complex. It can be done as `complex operator+(complex &c1, complex &c2)`
Please note that we are changing the value of `c2` in the operator function and hence option b is not correct.
This question is intentionally made as MSQ

Programming Questions

Question 1

Consider the program below.

- Fill in the blank at `LINE-1` to complete the function header.

The program must satisfy the given test cases. *Marks: 3*

```
#include <iostream>
#include <string>
using namespace std;
_______________________________ { // LINE-1
    string r = b + " " + a;
    cout << r;
}
int main() {
    string p;
    cin >> p;
    if (p == "x" || p == "X")
        print("Program");
    else
        print("Program", p);
    return 0;
}
```

Public 1

```
Input: C
Output: C Program
```

Public 2

```
Input: X
Output: Any Program
```

Private

```
Input: C++
Output: C++ Program
```

Answer:
`LINE-1: void print(string a, string b = "Any")`
Explanation:
The function header should have two parameters. The second parameter should have the default value "Any". So, `LINE-1` can be filled with
`void print(string a, string b = "Any")`

Question 2

Consider the following program.

- Fill in the blank at `LINE-1` with the appropriate parameter,
- Fill in the blank at `LINE-2` with the appropriate return statement

The program must satisfy the sample input and output. *Marks: 3*

```
#include <iostream>
using namespace std;
int Fun(_________________) { // LINE-1
    return ________; // LINE-2
}
int main() {
    int x, y;
    cin >> x >> y;
    cout << Fun(x + y);
    return 0;
}
```

Public 1

```
Input: 2 3
Output: 25
```

Public 2

```
Input: 4 -7
Output: 9
```

Private

```
Input: 5 -6
Output: 1
```

Answer:
`LINE-1:` `const int &x`
`LINE-2:` `x*x`
Explanation:
The function call is made with an argument which is a constant expression. As the function is called with a call-by-reference strategy, and actually an expression (x+y) is passed, the argument should be constant in nature. So, `LINE-1` should be filled as `const int &x`. The function gives a square value as output. So `LINE-2` should be filled as `x*x`.

Question 3

Consider the program below.

- Fill in the blank at `LINE-1` to complete the function header
- Fill in the blank at `LINE-2` to complete the return statement

The program must satisfy the given test cases. *Marks: 3*

```
#include <iostream>
using namespace std;
struct point {
    int x, y;
};
------------------------------ { //LINE-1
    pt.x += t;
    pt.y -= t;
    _____________; //LINE-2
}
int main() {
    int a, b, c;
    cin >> a >> b >> c;
    point p = {a, b};
    int t = c;
    point np = p + t;
    cout << "(" << np.x << ", " << np.y << "), (" << p.x << ", " << p.y  << ")";
    return 0;
}
```

Public 1

```
Input: 4 7 3
Output: (7, 4), (7, 4)
```

Public 2

```
Input: 10 20 5
Output: (15, 15), (15, 15)
```

Private

```
Input: 8 4 6
Output: (14, -2), (14, -2)
```

Answer:

```
LINE-1:  point operator+(point &pt, int t)
LINE-2:  return pt
```

Explanation:

We need to overload the addition operator for the structure `point` and integer. Also, we are changing the values of the passing parameter and the changed value should be reflected in the actual parameter. Hence, `LINE-1` should be filled as

```
point operator+(point &pt, int t)
```

and the return statement should be

```
return pt
```