Programming in Modern C++: Assignment Week 2

Total Marks : 20

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

July 12, 2023

Question 1

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

```
#include <iostream>
using namespace std;
int add(int n1 = 0) { return n1; }
int add(int n1 = 0, int n2 = 0) { return n1 + n2 - 1;}
int add(int n1 = 0, char c1 = 'a'){ return n1 + c1 - 1; }
int add(int n1 = 0, int n2 = 0, int n3 = 0) { return n1 + n2 + n3 - 1; }
int main() {
    int c = add(2, 5);
    cout << c << endl;
    return 0;
}
```

What will be the output/error(s)?

a) `6`

b) `99`

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

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

Answer: d)
Explanation:
The call `add(2, 5)`, can invoke both the following overloads of `add` function:
`int add(int n1 = 0, int n2 = 0) { ... }`
`int add(int n1 = 0, int n2 = 0, int n3 = 0) { ... }`
Thus, the call is ambiguous. Hence, the correct option is d.

Question 2

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

```
#include <iostream>
using namespace std;
#define Times(x, y) (x * y)

int main() {
    int a = 4;
    cout << Times(a + 3, 5) << endl;
    return 0;
}
```

What will be the output?

a) `35`

b) `20`

c) `19`

d) `23`

Answer: c)
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 << Times(a + 3, 5) << endl;
cout << (a + 3 * 5) << endl;
cout << (4 + 15) << endl;
cout << 19 << endl;
```

Hence, the correct option is c).

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 + 5;          //LINE-2
    i = i1;              //LINE-3
    i1 = 4+5;            //LINE-4
    return 0;
}
```

Which line/s will give compilation error/s?

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 code segment given below. *[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+1) = y;
    cout << x << " " << z;
    return 0;
}
```

What will be the output/error?

a) `6 6 2 2`

b) `7 7 3 3`

c) `6 6 3 2`

d) `Compilation error: invalid initialization of non-const reference of type 'int&' from an rvalue of type 'int'`

Answer: d)
Explanation:
The function `func` has non-const reference but we are passing `(x+1)` which is of constant expression. Hence it generates compilation error.

Question 5

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

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

What will be the output?

a) `5 30`

b) `30 25`

c) `25 36`

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 6

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

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

Which line will generate compilation error?

a) `LINE-1`

b) `LINE-2`

c) `Both LINE-1 and LINE-2`

d) `No error`

Answer: b)
Explanation:
The pointer `ptr` is of constant type but `b` is not a constant type pointer. Hence, assignment of non-const pointer to a constant pointer is illegal which generates compilation error.

Question 7

Consider the code segment below. *[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` so that the output is `-1 + i7`.

a) `complex operator-(const complex &c1, complex &c2)`

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

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

d) `complex operator-(const complex &c1, const complex &c2)`

Answer: a), c)
Explanation: The operator overload function is changing the value of the second parameter within the function. So, the second parameter of the function should not be constant. Hence, correct options are a) and c).

Question 8

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

```
#include <iostream>
using namespace std;
char fun(int a, char b){
    char c = a+b;
    return c;
}
char fun(char a, int b){
    char c = a-b;
    return c;
}
int main(){
    cout << fun(100,10); //LINE-1
    return 0;
}
```

What will be the output/error?

a) n

b) Z

c) NULL

d) `Compilation error at LINE-1:  call of overload 'fun(int, int)' is ambiguous`

Answer: d)
Explanation: The calling of function `fun(.)` is ambiguous because it matches with both the function prototype. Hence, the correct option is d).

Question 9

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

```
#include <iostream>
using namespace std;
#define MAXVAL(X, Y) (X > Y ? X : Y)
inline int maxval(int x, int y){
    return x > y ? x : y;
}
int main(){
    int x1 = 3, x2 = 4;
    cout << MAXVAL(++x1, ++x2) << " ";
    cout << maxval(++x1, ++x2) << " ";
    return 0;
}
```

What will be the output?

a) `5 6`

b) `6 6`

c) `6 5`

d) `6 7`

Answer: d)
Explanation:
In macro call, the arguments get substituted blindly, and then evaluated. However, in inline function call, the arguments get evaluated before those are passed to the function.
The call `MAXVAL(++x1, ++x2)` will be expanded as: `(++x1 > ++x2 > ++x1 : ++x2)` =¿ `(4 > 5 > ++x1 : ++x2) => (++x2) => 6`.
The call `maxval(++x1, ++x2) => maxval(5, 7)`, will return `7`.

Programming Questions

Question 1

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

- at `LINE-1` with function header

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

```
#include <iostream>
using namespace std;
_____________________ { // LINE-1
    int r = b + a;
    cout << r;
}
int main() {
    int a, b;
    cin >> a >> b;
    if (b < 0)
        print(a);
    else
        print(a,b);
    return 0;
}
```

Public 1

```
Input: 2 3
Output: 5
```

Public 2

```
Input: 8 -2
Output: 8
```

Private

```
Input: 4 -4
Output: 4
```

Answer:
`LINE-1: void print(int a, int b=0)`
Explanation:
The function header should have two parameters. The second parameter should have the default value 0. So, `LINE-1` can be filled with
`void print(int a, int b = 0)`

Question 2

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

- at `LINE-1` with appropriate function parameter,
- at `LINE-2` with appropriate statement

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

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

Public 1

```
Input: 4
Output: 16 16
```

Public 2

```
Input: 2
Output: 4 4
```

Private

```
Input: 5
Output: 25 25
```

Answer:
`LINE-1:  int &x`
`LINE-2:  x * x`
Explanation:
The function call is made with an argument which should not be constant. As the function is called with a call-by-reference strategy, the actual parameter value will be changed within the function. So, `LINE-1` should be filled as `int &x`. The function gives a square value as output. So `LINE-2` should be filled as `x*x`.

Question 3

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

- at `LINE-1` with appropriate function header,
- at `LINE-2` with appropriate return statement

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

```
#include <iostream>
using namespace std;
struct point {
    int x, y;
};
_______________________________ { //LINE-1
    pt1.x += pt2.x;
    pt1.y += pt2.y;
    _______________;    //LINE-2
}
int main() {
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    point p1 = {a, b};
    point p2 = {c, d};
    point np = p1 + p2;
    cout << "(" << p1.x << ", " << p1.y << ")" << "(" << np.x << ", " << np.y << ")";
    return 0;
}
```

Public 1

```
Input: 1 2 3 4
Output: (4, 6)(4, 6)
```

Public 2

```
Input: 2 4 6 8
Output: (8, 12)(8, 12)
```

Private

```
Input: 2 1 -2 -1
Output: (0, 0)(0, 0)
```

Answer:
`LINE-1:  point operator+(point &pt1, point &pt2)`
`LINE-2:  return pt1`
Explanation:
We need to overload the addition operator for the structure `point`. 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 &pt1, point &pt2)`
and the return statement should be
`return pt1`