Programming in C++: Programming Test-1

Total Marks : 20

September 21, 2017

Question 1

Consider the following code. Perform the necessary modifications in the code (in editable section), so that the calling sequence of the constructor and destructor would be: B(), D(), ~D() and ~B().
Note: Don't edit/ Modify the `cout` statements. *Marks 2*

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

class B {
public:
    B() { cout << "98-"; }
    ~B() { cout << "56"; }  // Do not Edit/Modify the "cout"
};

class D : public B {
    int n;
public:
    D(int p):n(p) { cout << n << "-"; }
    ~D() { cout << n*2 << "-"; }
};

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

    B *basePtr = new D(n);

    delete basePtr;

    return 0;
}
```

Public test case-1

Input: 2
Output: 98-2-4-56

Public test case-2

Input: 8
Output: 98-8-16-56

Private test case-1

Input: 3
Output: 98-3-6-56

Solution:

```
virtual ~B()
```

Question 2

The program indicates the concept of mutability. Fill the blank with appropriate keywords to satisfy the given test cases. *Marks 2*

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

class MyClass {
    int mem_;
    __________ int x_;
public:
    MyClass(int m, int mm) : mem_(m), x_(mm) {}

    int getxMem() __________ { return x_; }
    void setxMem(int i) __________ { x_ = i; }
};

int main() {
    int x, y, z;

    cin >> x;
    cin >> y;
    cin >> z;

    const MyClass myConstObj(x, y);

    myConstObj.setxMem(z);
    cout << myConstObj.getxMem() << endl;

    return 0;
}
```

Public Test case-1

Input: 5 7 8
Output: 8

Public Test case-2

Input: 0 1 0
Output: 0

Private Test case-1

Input: 11 11 11
Output: 11

Solution

```
mutable // const // const
```

Question 3

Write the required keywords and function names to get the output as per the test cases.

Marks 2

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

____________ NS {
    int abs(int n) {
        if (n < -128) return 0;
        if (n > 127) return 0;
        if (n < 0) return -n;
        return n;
    }
}

int main() {
    double x, y, z;

    cin >> x;
    cin >> y;
    cin >> z;

    cout << _______(x) << " "
    << _______(y) << " "
    << _______(z) << " "
     <<  endl;

    cout <<_______(x) << " "
    << _______(y) << " "
    << _______(z) << " "
    << endl;

    return 0;
}
```

Public Test case-1

Input: -203, -69, 9

```
Output: 0   69 9
0 69 9
```

Public Test case-2

Input: 178, 45, 0

```
Output: 0   45 0
0 45 0
```

Private Test case

Input: -114, 20, 2

```
Output: 114  20  2
```

114 20 2

Solution

```
namespace // NS::abs(x) // NS::abs(y) // NS::abs(z) //  abs(x)//  abs(y) // abs(z)
```

Question 4

Fill in the blank below by writing the appropriate operator function, parameters and return type so that the given test cases will be satisfied. *Marks 2*

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

class Complex { double re, im; public:
    explicit Complex(double r = 0, double i = 0) : re(r), im(i) { }

    void disp() { cout << re << "+j " << im << endl; }

    friend Complex operator+ (const Complex &a, const Complex &b) {
        return Complex(a.re + b.re, a.im + b.im);
    }
    friend Complex operator+ (const Complex &a, double d) {
        Complex b(d); return a + b;
    }
    ___________________________ {
        Complex a(d); return a + b;
    }
};

int main(){
    double x, y, z, w;

    cin >> x >> y;
    cin >> z >> w;

    Complex d1(x, z), d2(y, w), d3;

    d3 = d1 + d2; d3.disp();
    d3 = d1 + 6.2; d3.disp();
    d3 = 4.2 + d2; d3.disp();

    return 0;
}
```

Public Test case-1

Input:

```
3.4 5.6
6 7
```

Output:

```
Output:
9 + j 13
9.6 + j 6
9.8 + j 7
```

Public Test case-2

Input:

```
5 7
4 5
```

Output:

```
Output:
12 + j 9
11.2 + j 4
11.2 + j 5
```

Private Test case

Input:

```
0 1
1 1
```

Output:

```
Output:
1 +j 2
6.2 +j 1
5.2 +j 1
```

Solution

```
friend Complex operator+ (double d, const Complex &b)
```

Question 5

Consider the class 'Box' where dimensions are defined. complete the code so that two boxes with different dimensions can be added/merged to form another new box with new dimensions.

Marks: 2

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

class Box {
private:
    int length, breadth, height;
public:

    Box(int a = 0, int b = 0, int c = 0) :
    length(a), breadth(b), height(c) {};

    int getDimension() {
        return length + breadth + height;
    }

    Box ___________(const Box& b){ // Fill the name of the function
        Box box;
        ____________________ // Fill the statement
        ____________________ // Fill the statement
        ____________________ // Fill the statement

        return box;
    }
};

int main() {
    int l = 0, b = 0, h = 0;
    cin >> l; cin >> b; cin >> h;

    Box Box1(4, 6, 8), Box2(l, b, h), Box3;
    Box3 = Box1 + Box2;

    int dim = Box3.getDimension();
    cout << dim;

    return 0;
}
```

Public Test case-1

```
Input:
6
4
2
Output:
20
```

Public Test case-2

```
Input:
1
1
10
Output:
20
```

Private Test case

```
Input:
6
6
6
Output:
20
```

Solution

Explanation: The addition of length, breadth, height of new box, is taking place in the function getDimension(). But in the addition function i.e operator+(), you need to set the dimensions of new box by comparing the two boxes and choosing the max value. which will set the dimension of new box ; length = 4,breadth=6 and height = 8 always, regardless of your I/P value.

```
operator+ //

box.length = max(length, b.length); // Fill the statement
box.breadth = max(breadth, b.breadth); // Fill the statement
box.height = max(height, b.height); // Fill the statement
```

Question 6

Fill up the blanks to get the desired output according to the test cases. *Marks 2*

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

class Complex { private: double re_, im_;
public:
     Complex(double re = 4.0, double im = 5.0): re_(re), im_(im)
         { cout << "Ctor: (" << re_ << ", " << im_ << ")" << endl; }
     ~Complex()
         { cout << "Dtor: (" << re_ << ", " << im_ << ")" << endl; }

     void print() { cout << "|" << re_ << "+j" << im_ << "| " << endl; }
};

____________________;


int main() {
    cout << "main" << endl;
    double x, y;

    cin >> x;

    cin >> y;
    Complex d(x); Complex e;
    c.print();
    d.print();

    return 0;
}
```

Public Test case-1

Input: 5 6
Output:

```
Ctor: (8, 4)
main
Ctor: (5, 5)
Ctor: (4, 5)
|8+j4|
|5+j5|
Dtor: (4, 5)
Dtor: (5, 5)
Dtor: (8, 4)
```

Public Test case-2

Input: 2.5 3.5
Output:

```
Ctor: (8, 4)
main
Ctor: (2.5, 5)
Ctor: (4, 5)
|8+j4|
|2.5+j5|
Dtor: (4, 5)
Dtor: (2.5, 5)
Dtor: (8, 4)
```

Private Test case-1

Input: 0 1
Output:

```
Ctor: (8, 4)
main
Ctor: (0, 5)
Ctor: (4, 5)
|8+j4|
|0+j5|
Dtor: (4, 5)
Dtor: (0, 5)
Dtor: (8, 4)
```

solution

Complex c(8, 4)

Question 7

Consider the code given below. Fill in the blank to complete the code to match the output of the test cases. *Marks: 2*

Public

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

class MyClass {
    static int x;
public:

    void get() { x++; }
    ---------- --------------- print(int y) {
        x = x - y;
        cout << x << " ";
    }
};

------------------------------;

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

    MyClass::print(x);

    MyClass o1;
    o1.get();
    o1.print(x);

    return 0;
}
```

Public-1

```
Input:
5
Output: -4  -8
```

Public-2

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

Private

```
Input:
-7
Output: 8 16
```

Solution

```
static void // int MyClass::x = 1;
```

Question 8

Write the required constructor and function definitions of the class Stack to get the output as per the test cases. *Marks: 2*

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

class Stack {
    ______________________: // Write the appropriate Access specifier
    vector<char> data_; int top_;

public:
    int empty() { _________________________; }
    void push(char x) { _________________________; }
    void pop() { ________________; }
    char top() { ___________________; }
};
int main() {
    Stack s;
    char str[20];

    cin >> str;

    s.data_.resize(100);
    s.top_ = -1;

    for(int i = 0; i < strlen(str) ; ++i)
        s.push(str[i]);

    while (!s.empty()) {
        cout << s.top(); s.pop(); s.pop();
    }

    return 0;
}
```

Public 1

```
Input:
erty
Output: yr
```

Public 2

```
Input:
ghjilk
Output: kih
```

Private 1

```
Input:
```

ADAM
Output: MD

Solution

```
public // return (top_ == -1) // data_[++top_] = x // --top_ // return data_[top_]
```

Question 9

Fill in the blanks below to complete the program. The inputs and the desired output are given in form of test cases. *Marks: 2*

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

class Date {
    int da;  // 2 digit day
    int mo;  // 2 digit month
    int yr;  // 4 digit year
public:
    Date(int d, int m, int y): da(d), mo(m), yr(y) { }

    friend ________ operator<<(ostream& os, const Date&); // Fill the return type
};

________ operator<<(ostream& os, const Date& dt) // Fill the return type
{
    ________________________ // Fill the implementation

    return os;
}

int main() {
    int day = 0, month = 0, year = 0;

    cin >> day;
    cin >> month;
    cin >> year;

    Date dt(day, month, year);

    cout << dt;

    return 0;
}
```

Public 1

```
Input:
25
10
2015
Output:
25/10/2015
```

Public 2

```
Input:
12
11
2013
```

```
Output:
12/11/2013
```

Private 1

```
Input:
3
3
3
Output:
3/3/3
```

Solution

```
 ostream&  // ostream& // os << dt.da << "/" << dt.mo << "/" << dt.yr;
```

Question 10

Consider the code given below. Fill up the marked lines to complete the code to match the output of the test cases. *Marks: 2*

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

class A {
protected:
    int n;
    A(int i) : n(i) { }
    virtual void print() = 0;
    virtual int get(){ return n+1; }
};

class B : private A {
public:
    B(int i) : A(i) {}

    int get() {
        ____________________ // The get function body
    }

};

class C : public B {
public:
    C(int i) : B(i) {}
    void print() {
       cout <<___________<< endl; // display the result of the get function
    }
};

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

    C *p = new C(n);
    p->print();

    return 0;
}
```

Public 1

```
Input:
9
Output:
9
```

Public 2

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

Private 1

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

Solution

```
return n;
get()
```