Programming in C++

Total Marks : 20

September 20, 2017

Question 1

Fill in the blanks below with appropriate C++ keyword and statements so that the given test cases are satisfied. *Marks 2*

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

class MyClass {
public:
    _________  int count; // Fill the blank with correct keyword

    MyClass() { count++; }
    ~MyClass() { count--; update(); }

    void update() { count *= 2; }
};

_____________________ // Definition and initialization

int main() {

    cin >> MyClass::count;

    MyClass *pt;
    pt = new MyClass[3];

    _____________________ // Delete the objects

    cout << MyClass::count;

    return 0;
}
```

Public-1

```
Input:
22

Output: 186
```

Public-2

```
Input:
102

Output: 826
```

Private

```
Input:
200

Output: 1610
```

Answer

```
static
int MyClass::count = 0;
delete[] pt;
```

Question 2

Fill in the blanks below to satisfy the given test cases. Follow the instructions in the comments.

Marks 2

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

class Polygon {
protected:
    int width, height;
public:
    Polygon(int a, int b) : width(a), height(b) {}

    // Declare the area function here
    ------------------------------------------------

    void printarea() { cout << this->area() << ":"; }
};

class Rectangle : public Polygon {
public:
    // Declare the constructor to initialize width and length of rectangle
    ------------------------------------------------------

    int area() { return width*height; }
};

class Triangle : public Polygon {
public:
    Triangle(int a, int b) : Polygon(a, b) {}
    int area() { return width*height / 2; }
};

int main() {
    int h, w;
    cin >> h >> w;

    // Declare ppoly1 and ppoly2 as "pointer to Polygon" and dynamically allocate
    // a Rectangle and a Triangle objects respectively held by these pointers

    ------------------------------------ // Rectangle object of h and w
    ------------------------------------ // Triangle object of h and w

    ppoly1->printarea();  // Print area of Rectangle
    ppoly2->printarea();  // Print area of Triangle

    delete ppoly1;
    delete ppoly2;

    return 0;
}
```

Public-1

```
Input:
10 20

Output: 200:100:
```

Public-2

```
Input:
4 8

Output: 32:16:
```

Private

```
Input:
100 200

Output: 20000:10000:
```

Answer

```
virtual int area(void) = 0;
Rectangle(int a, int b) : Polygon(a, b) {}
Polygon *ppoly1 = new Rectangle(h, w);
Polygon *ppoly2 = new Triangle(h, w);
```

Question 3

Fill in the blanks below to satisfy the given test cases. Follow the instructions in the comments.

Marks 2

Do not change any other part of the code.

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

class Area {
public:
    int calc(int l, int b) { return l*b; }
};

class Perimeter {
public:
    int calc(int l, int b) { return 2 * (l + b); }
};

/* Rectangle class is derived from classes Area and Perimeter. */
class Rectangle: ________________________ { // Inherit the required base classes
private:
    int length, breadth;
public:
    Rectangle(int l, int b) : length(l), breadth(b) {}

    int area_calc() {
    /* Calls calc() of class Area and returns it. */
        ------------------------------------------
    }
    int peri_calc() {
    /* Calls calc() function of class Perimeter and returns it. */
        ------------------------------------------
    }
};

int main() {

    int l, b;
    cin >> l >> b;  // Read variables l and b from the keyboard
    Rectangle r(l, b);   // Create Rectangle object r

    cout << r.area_calc() << endl;
    cout << r.peri_calc();

    return 0;
}
```

Public-1

```
Input:
6 3

Output: 18
        18
```

Public-2

```
Input:
4 8

Output: 32
        24
```

Private

```
Input:
101 201

Output: 20301
        604
```

Answer

```
public Area, Perimeter
return Area :: calc(length, breadth);
return Perimeter :: calc(length, breadth);
```

Question 4

Fill in the blanks below to satisfy the given test cases. Follow the instructions in the comments.

Marks 2

Do not change any other part of the code.

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

class Complex {
    int real; int imag;
public:
    Complex(int a = 0, int b = 0): real(a), imag(b) {}

    // Write function header and body for
    // the copy-assignment operator
    ---------------------------------- {

           --------------------

           --------------------

           --------------------
    }

    void print() {
        cout << real << "+i" << imag;
    }
};

int main() {

    int a = 0, b = 0;
    cin >> a >> b;

    Complex t1(a, b);
    Complex t3;
    t3 = t1;     // Using copy-assignment operator
    t3.print();

    return 0;
}
```

Public-1

```
Input:
8 9

Output: 8+i9
```

Public-2

```
Input:
15 10

Output: 15+i10
```

Private

```
Input:
22 32

Output: 22+i32
```

Answer

```
Complex& operator=(const Complex &c)
real = c.real;
imag = c.imag;
return *this;
```

Question 5

Consider the following code. Fill up the code so that `print()` can print the value of `A::n` and match the test cases. *Marks 2*

Do not change any other part of the code.

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

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

int A::get() {
    return n;
}

class B : private A {
protected:
    B(int i) : A(i) {}
    //------------------------- Fill your code here----------------------




    //------------------------------------------------------------------
};

class C : public B {
public:
    C(int i) : B(i) {}

    void print() {
        cout << get() << endl;
    }
};

int main() {

    int n;
    cin >> n;

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

    return 0;
}
```

Public-1

```
Input:
10

Output: 10
```

Public-2

```
Input:
22

Output: 22
```

Private

```
Input:
8

Output: 8
```

Answer

```
int get() {
    return A::get();
}
```

Question 6

Fill in the blanks below to satisfy the given test cases. Follow the instructions in the comments.

Marks 2

Do not change any other part of the code.

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

/* Write the header and body of generic function display() which takes
a single generic parameter and prints its value followed by a blank */

// ---------------------Write the code here ------------------


//-------------------------------------------------------------


/* Write an overload of generic function display() which takes two
generic parameters and prints the values with a space between them
and a new line at the end */

// ----------------------- Write code here ----------------------



//-----------------------------------------------------------------

int main() {

    double d;
    int i;
    char c;

    cin >> i;
    cin >> d;
    cin >> c;

    display(c);
    display(i, d);
    display(c, d);

    return 0;
}
```

Public-1

```
Input:
10 12.2 c

Output: c 10 12.2
        c 12.2
```

Public-2

```
Input:
8 8 a

Output: a 8 8
        a 8
```

Private

```
Input:
2 10.8 g

Output: g 2 10.8
        g 10.8
```

Answer

```
template <class T>
void display(T x) {
    cout << x << " ";
}

template <class T1, class T2>
void display(T1 x, T2 y) {
    cout << x << " " << y << endl;
}
```

Question 7

Fill in the blanks below to satisfy the given test cases. Follow the instructions in the comments.

Marks 2

Do not change any other part of the code.

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

class A {
    int a, b;
public:
    A(int i, int j): a(i), b(j) { }

    ________________________ // Declare class B as Friend of class A
};

class B {
public:
    int dispSum(A& x) {
        ____________________ // Find sum of a and b members of x
                             // and return the result
    }
};

int main() {

    int i, j;
    cin >> i >> j;
    A a(i, j);
    B b;

    cout << b.dispSum(a);

    return 0;
}
```

Public-1

```
Input:
10 20

Output: 30
```

Public-2

```
Input:
2 5

Output: 7
```

Private

```
Input:
8 2

Output: 10
```

Answer

```
friend class B;

return (x.a + x.b) ;
```

Question 8

Write an appropriate constructor or an operator function in the following program to cast a `class A` object to a `char *` object. *Marks 2*

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

class A {
public:
    char *str;
    A(char *s) : str(s) { }

// ---- Write the appropriate constructor or
// conversion operator function below ------




//-----------------------------------------
};

int main() {

    char input[20];
    cin >> input;
    A a(input);

    // A ==> char *
    char *s = static_cast<char*>(a);
    strcat(s, "_smartphone");
    cout << s;

    return 0;
}
```

Public-1

```
Input:
samsung

Output: samsung_smartphone
```

Public-2

```
Input:
apple

Output: apple_smartphone
```

Private

```
Input:
nokia

Output: nokia_smartphone
```

Answer

```
operator char *() {
    return (str);
}
```

Question 9

Consider the following code. Write the correct `swap` function to match the test cases. *Marks 2*

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

// ----------------- Write the Swap function here ----------------------
// --------- You cannot write more than one Swap function -------------



// ---------------------------------------------------------------------

int main() {

    int a, b;
    double s, t;
    string mr, ms;

    cin >> a >> b;
    cin >> s >> t;
    cin >> mr >> ms;

    Swap(a, b);
    Swap(s, t);
    Swap(mr, ms);

    cout << a << " " << b << " ";
    cout << s << " " << t << " ";
    cout << mr << " " << ms;

    return 0;
}
```

Public-1

```
Input:
5 2 11.66 3.3 come wel

Output: 2 5 3.3 11.66 wel come
```

Public-2

```
Input:
20 30 2.2 3.3 hello world

Output: 30 20 3.3 2.2 world hello
```

Private

```
Input:
9 15 77.7 88.8 program c++

Output: 15 9 88.8 77.7 c++ program
```

Answer

```
template <typename T>
void Swap(T& x, T& y) {
    T tmp = x;
    x = y;
    y = tmp;
}
```

Question 10

Consider the following code. Insert proper codes in marked lines to match the test cases.

Marks 2

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

class myexception : ___________ { // Inherit standard exception with
                                  // appropriate visibility

    virtual const char* what() const throw() {
        return "DivideByZero";
    }
};

class DivideByZero {
        int numerator, denominator;
    public:
        DivideByZero(int a = 0, int b = 0) : numerator(a), denominator(b) {}
        int divide(int numerator, int denominator) {
            if (denominator == 0) {
                ____________ // Raise exception suitably to handle
                             // divide by zero error
            }
            return numerator / denominator;
        }
};

int main() {

    DivideByZero d;
    int a, b;
    cin >> a >> b;

    try {
        d.divide(a, b);
    }

    catch (exception& e) {
         cout << e.what() << endl;
    }

    return 0;
}
```

Public-1

```
Input:
20 0

Output: DivideByZero
```

Public-2

```
Input:
3.3 0

Output: DivideByZero
```

Private

```
Input:
10 0

Output: DivideByZero
```

Answer

```
public exception
throw myexception()
```