Programming in C++: Assignment Week 6

Total Marks : 20

August 26, 2017

Question 1

```
class A { public:
    virtual void f(int) { }
    virtual void g(double) { }
    virtual void d(char) { }
    int h(A *) { }
};

class B: public A { public:
    void f(int) { }
    virtual int h(B *) { }
};

class C: public B { public:
    void g(double) { }
    void d(char) { }
    int h(B *) { }
};

A *a = 0;
a->d(s);
```

What will be the symbolic expression that a− >d(s); compiles to? *Marks 2*

a) a− >vft[2](a, s);

b) a− >vft[1](a, s);

c) Error

d) C::d(a, s);

Answer: a)
Explanation: Concept of virtual function table. Please refer to the course video lectures for more details.

Question 2

Identify the abstract classes from the following Code snippet. *Marks 2*

```
class Vehicle {
    public:
        virtual void drive() = 0 { cout << "Vehicle"; }
};

class LandVehicle: public Vehicle {
    void drive() { cout << "Land Vehicle";}
};

class AirVehicle: public Vehicle {
};

class Car : public LandVehicle {
    public:
        void drive() { cout << "Car"; }
};

class Truck : public LandVehicle {
    public:
        void drive() { cout << "Truck";}
};

class Aeroplane : public AirVehicle {
    public:
        void drive() { cout << "Aeroplane";}
};

class Indigo : public Aeroplane {
};
```

a) Vehicle, LandVehicle, AirVehicle

b) Vehicle, LandVehicle, AirVehicle, Aeroplane

c) Vehicle, AirVehicle

d) Vehicle

Answer: c)
Explanation: Abstract base classes are those classes which contains only pure virtual functions. Moreover a class derived from an abstract base class will also be abstract unless you override each pure virtual function in the derived class

Question 3

How many virtual table will be set up by the compiler for the following program? *Marks 2*

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

class Base {
    public:
        virtual void function1() {};
        virtual void function2() {};
        virtual void function3() {};
};

class D1: public Base {
    public:
        virtual void function1() {};
};

class D2: public Base {
    public:
        virtual void function2() {};
        virtual void function1() {};
};

class D3: public Base {
    public:
        virtual void function3() {};
};
```

a) 2

b) 1

c) 4

d) 3

Answer: c)
Explanation: Depends on the total number of polymorphic classes. Refer course video lectures for more details.

Question 4

What will be the output of the following program? *Marks 2*

```
#include<iostream>
using namespace std;
class Base {
    protected:
        double var;
    public:
        virtual void fun() = 0;
        Base(double i) { var = i; }
};
class Derived: public Base {
    double dervar;
    public:
        Derived(double i, double j):Base(i) { dervar = j; }
        void fun() { cout << "var = " << var << ", dervar = " << dervar; }
};
int main(void) {
    Derived d(14.6, 8);
    d.fun();
    return 0;
}
```

a) Compilation Error: Type mismatch for arguments passed

b) var = 14.6, dervar = 8

c) Compilation Error: Invalid access in Constructor

d) Compilation Error: Undefined reference of fun()

Answer: b)
Explanation: Overridden virtual functions can be called with the respective class object instance. Static binding.

Question 5

What will be the output of the following program? *Marks 2*

```
#include <iostream>
using namespace std;
class A { public:
    void f() { cout << "A::f()" << endl; }
    virtual void g() { cout << "A::g()" << endl; }
    void h() { cout << "A::h()" << endl; }

};

class B : public A { public:
    void f() { cout << "B::f()" << endl; }
    void g() { cout << "B::g()" << endl; }
    virtual void h() { cout << "B::h()" << endl; }
};

class C : public B { public:
    void f() { cout << "C::f()" << endl; }
    void g() { cout << "C::g()" << endl; }
    void h() { cout << "C::h()" << endl; }

};

int main() {
    B *q = new C; A *p = q;
    p->f();
    p->h();

    q->f();
    q->h();
    return 0;
}
```

a) A::f()
   A::h()
   B::f()
   C::h()

b) A::f()
   B::f()
   A::h()
   C::h()

c) A::f()
   B::f()
   C::h()

A::h()

d) A::f()
A::h()
C::h()
B::f()

Answer: a)
Explanation: Concept of static and dynamic binding. Refer lecture slides for more details.

Programming Assignment

Question 1

Fill in the missing parts in the following program as per the instructions so that the given test cases will be satisfied. *Marks 3*

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

class Base {
    public:
        _________________________    // Make function show() as a pure virtual function
};

class Derived : public Base {
    int i;
    public:
        Derived(int num) : i(num) { i = i * 2; }
        _________________________     // Define show() function to print
                                      //  the value of d.i
};

int main() {

    int n;
    cin >> n;
    Derived d(n);
    Base &b = d;
    b.show();
    return 0;
}
```

Answer: virtual void show() = 0; // void show() { cout << i; }
Explanation: A child class overrides the pure virtual methods to become a concrete class

- Input: 3
  Output: 6
- Input: -2
  Output: -4
- Input: 20
  Output: 40

Question 2

Modify the code in editable section to match the test cases. Don't modifying any cout *Marks 3*

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

class B {
public:
    B() { cout << "98 "; } // don't modify the "cout"



    ~B() { cout << "56 "; }  // Don't 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;
}
```

Answer: virtual B() //
Explanation: If the destructor is not virtual in a polymorphic hierarchy, it leads to Slicing. So Destructor must be declared virtual in the base class.

- Input: 3
  Output: 98 3 6 56
- Input: 2
  Output: 98 2 4 56
- Input: 56
  Output: 98 56 112 56

Question 3

Consider the following program. Write the correct code in the editable section, so that print() can print the value of A::n and match the outputs of the test cases. *Marks 4*

```
#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:
        //------------------------ Template Code (Editable)----------------------
        // Define the constructor to initialize A::n
        // Override get() function to return A::n
        //------------------------ Suffix Fixed Code --------------------------
};

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;
}
```

Answer: B(int i) : A(i) // int get() return A::get();
Explanation: If class B does not override A::get(), class C cannot call get() because it is A::get() and class A being a private base class of B, will not allow access to class C (Bs child). So B::get() is needed. Naturally, it has to call A::get().

- Input: -10
  Output: -10

- Input: 20
  Output: 20
- Input: 1000
  Output: 1000