CS 274—Object Oriented Programming with C++
Final Exam

(8 points) 1. Show the output of the following program:

```
#include<iostream>
class Base {
public:
   Base(){cout<<"Base"<<endl;}
   Base(int i){cout<<"Base"<<i<<endl;}
   ~Base(){cout<<"Destruct Base"<<endl;}
};

class Der: public Base{
public:
   Der(){cout<<"Der"<<endl;}
   Der(int i): Base(i) {cout<<"Der"<<i<<endl;}
  ~Der(){cout<<"Destruct Der"<<endl;}
};
int main(){
  Base a;
  Der d(2);
  return 0;
}
```

(8 points) 2. Show the output of the following program:

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

class C {
  public:
       C(): i(0) { cout <<  i << endl; }
      ~C(){ cout << i << endl; }
       void iSet( int x ) {i = x; }
  private:
        int i;
};
int main(){
     C c1, c2;

     c1.iSet(5);

      {C c3;
         int x = 8;
         cout << x << endl;
        }
      return 0;
}
```

(8 points) 3. Show the output of the following program:

```
#include<iostream>

class A{
public:
  int f(){return 1;}
  virtual int g(){return 2;}
};

class B: public A{
public:
  int f(){return 3;}
  virtual int g(){return 4;}
};

class C: public A{
public:
  virtual int g(){return 5;}
};

int main(){
  A *pa;
  A a;
  B b;
  C c;

 pa=&a;  cout<<pa -> f()<<endl;  cout<<pa -> g()<<endl;
 pa=&b;  cout<<pa -> f() + pa -> g()<<endl;
 pa=&c;  cout<<pa -> f()<<endl;  cout<<pa -> g()<<endl;

  return 0;
}
```

(8 points) 4. Show the output of the following program:

```
#include<iostream>

class A{
protected:
   int a;
public:
   A(int x=1) {a=x;}
  void f(){a+=2;}
  virtual g(){a+=1;}
  int h() {f();  return a;}
  int j()  {g(); return a;}
};

class B: public A{
private:
  int b;
public:
   B(){int y=5){b=y;}
   void f(){b+=10;}
   void j(){a+=3;}
};

int main(){
   A obj1;
   B obj2;

   cout<<obj1.h()<<endl;
   cout<<obj1.g()<<endl;
   cout<<obj2.h()<<endl;
   cout<<obj2.g()<<endl;

  return 0;
}
```

(10 points) 5. Circle TRUE or FALSE for each of the following statements:

TRUE FALSE	An abstract base class cannot be instanced.
TRUE FALSE	Pointers to a base class may be assigned the address of a derived class object.
TRUE FALSE	A pure virtual method must be overridden in a derived class.
TRUE FALSE	An abstract base class cannot have non-abstract derived classes.
TRUE FALSE	The assignment operator may be overloaded as a method.
TRUE FALSE	Polymorphic functions only exist outside of inheritance hierarchies.
TRUE FALSE	A derived class cannot have a method with the same name as a base class method.
TRUE FALSE	If a binary operator is overloaded using a top-level function, then two parameters are required.
TRUE FALSE	A unary operator overloaded as a method still requires one parameter.
TRUE FALSE	A map is a sequential container.

(22 points) 6. Declare a class named Triple with three private data members (floats) x, y, and z. Provide public functions for setting and getting values of all the private data members. Define a constructor that initializes the values to user-specified values or, by default, sets the values all equal to 0. Also overload the following operators:

—Addition so that corresponding elements are added together

—Output so that it displays the Triple in the form “The triple is (x, y, z).”

—Assignment that copies x to z, y to x, and z to y.

—Post-increment so that x and z are increased by one each.

—Function call operator so that the values for x, y and z can be set.

(22 points) 7. Write a program that has an abstract base class named Quad. This class should have four member data variables (floats) representing side lenghts and a pure virtual function Area. It should also have a method for setting the data variables. Derive a class Rectangle from Quad and override the Area method so that it returns the area of the Rectangle. Write a *main* function that creates a Rectangle and sets the side lengths. Also write a top-level function that will take a parameter of type Quad and return the value of the appropriate Area function.

(14 points) 8. Write a template class Point with two class parameters representing the two coordinates of the Point. Include public methods to display and set the data values as well as a function that swaps the values so that, after the swap, the first element becomes the second and the second becomes the first. Also write a *main* function that creates a Point object and calls the public methods.