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

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;
}
```

2. 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.

3. 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.

4. Cross out all the lines that will not compile **in the main function** of the following program:

```
#include<iostream>

class A{
public:
   A(){}
   virtual int output() = 0;
private:
   int i;
};

class B: public A{
private:
    int j;
};

class C{
public:
   int f(int a){return x*a;}
protected:
   void setX(int a){x=a;}
   int getX(){return x;}
private:
   int x;
};

class D: public C{
private:
   int z;
};

int main(){
   A objA;
   B objB;
   C objC;
   D objD;

   C.setX(2);
   cout<<C.getX();

   D.setX(1);
   D.f(3);

   return 0;
}
```

5. 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;
}
```

6. 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 is cast into the second and the second is cast into the first. Also write a *main* function that creates a Point object and calls the public methods.

7. Write a class that contains two class data members *numBorn* and *numLiving*. The value of *numBorn* should be equal to the number of objects of the class that have been instanced. The value of *numLiving* should be equal to the total number of objects in existance currently (ie, the objects that have been constructed but not yet destructed.)

8. 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. |