ECS 40
Introduction to Software Development

Sample Midterm Examination

Name__________________________________________

Directions:

This is a closed-book test. Answer all questions on the paper provided. Use the backs of the provided paper if you need extra space.

1. What is the difference between a class and an instance of a class. Give an example.

2. What is information hiding, and how is it implemented in C++?

3. What is “operator overloading,” and how is it implemented in C++?

4. What is a friend function? What is the difference between a friend function and a member function of a class? Give an example of a friend function from your assignments in the class.

5. What are the operations of a stack? A queue?

6. Given the list of integers

   4, 7, 10, 3, 5, 9, 1, 6

   exhibit a binary tree that, when traversed in inorder, outputs the list in ascending order.
7. Given the class **Polynomial** defined by

```cpp
class Polynomial {
private :
    .
    .
    .
public :
    Polynomial()
    .
    .
    .
    Polynomial Sum ( Polynomial*, const int ) ;
}
```

and the following function that is a member function of the **Polynomial** class

```cpp
Polynomial Polynomial :: Sum ( Polynomial* array, const int n )
{
    Polynomial sum ;
    for ( int i = 0 ; i < n ; i++ ) {
        Polynomial vv = array[i] ;
        sum += vv ;
    }
    sum *= 1.0 / n ;
    return ( sum ) ;
}
```

State specifically where the constructors, destructors, copy constructors and assignment (operator=) for the **Polynomial** class are executed in this function. Be specific on how many of each are executed and where they are executed. Also state if any other member functions of the **Polynomial** class are executed in this routine.
8. What is a complete class in C++? What functions or operators does this class contain?

9. What are the differences in the function calls between the four member functions of the Shape class below. Be explicit!

```cpp
void Shape :: member ( Shape s1, Shape s2 ) ;
void Shape :: member ( Shape *s1, Shape *s2 ) ;
void Shape :: member ( Shape& s1, Shape& s2 ) const ;
void Shape :: member ( const Shape& s1, const Shape& s2 ) ;
void Shape :: member ( const Shape& s1, const Shape& s2 ) const ;
```