Your Name: _____ TA's Name: _____
ID#: _____ Section #: _____

SOLUTION: Take-Home: Quiz 8 (15 pts) – More OOP!

1. **(8 pts – 2 pts/each)** What are the Four Pillars of Object-Oriented (OOP) ? Explain each using your own words.

1. **Abstraction:**

   Abstraction **is the concept that only essential information is presented to the outside word, while the background details are hidden. There are many levels of abstraction.**

   Procedural abstraction **refers to a sequence of steps/instructions that have a specific and limited function. The name of the procedure implies these functions, but suppresses the specific details. An example would be the word "open" for a door. (Pressman – Software Engineering)**

   Data abstraction **is a named collection of data that describes a data object. An example would be the data abstraction "door". The "door" would represent a set of attributes like door type, weight, etc. (Pressman – Software Engineering)**

2. **Encapsulation:**

   The grouping of data or attributes and the operations or functions that can be applied to the data or attributes. Encapsulation provides the ability to restrict access to the data and hide information.

3. **Inheritance:**

   The ability of a class (derived) to derive properties from a previously defined (base) class. The derived class customizes the properties of the base or acquired class. It's considered a form of software or code reuse.

4. **Polymorphism:**

   Polymorphism **is the ability to use the same expression to denote different operations.** ***Runtime polymorphism*** **is the ability to associate multiple meanings to a single function name though the use of late or dynamic binding. You can process objects of the same class hierarchy as if they are all objects of the hierarchy's** ***base*** **class.** ***Compile time polymorphism*** **is the type that is achieved through function overloading, operator overloading, and templates. Polymorphism enables you to**

Your Name: _____ TA's Name: _____
ID#: _____ Section #: _____

"program in the general", instead of "program in the specific". Another form is *parametric* polymorphism the (data) type is left unspecified and later instantiated templates provide parametric polymorphism.

2. **(3 pts)** Given the following function, what is the worst-case Big-O time complexity?

_____O(n^3)_____

```
// Prints all subarrays in arr[0..n-1]
void subArray(int arr[], int n)
{
    // Pick starting point
    for (int i=0; i <n; i++)
    {
        // Pick ending point
        for (int j=i; j<n; j++)
        {
            // Print subarray between current starting
            // and ending points
            for (int k=i; k<=j; k++)
            {
                cout << arr[k] << " ";
            }

            cout << endl;
        }
    }
}
```

3. **(4 pts)** What is the difference between a *class* and a *struct*? Explain in your own words. Also, be sure to discuss the visibilities of members in each.

There's generally a misconception that a struct cannot apply many of the same features that a class can. However, a struct has many of the same properties of a class with some exceptions. All of the members of a *struct* are by default *public* and all of the members of a *class* are by default *private*. When deriving a struct from another struct or class, the access-specifier for a base struct or class is *public* by default. When deriving a class from another class, the access-specifier is *private* by default.