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

Using Canvas <https://canvas.wsu.edu/>, please submit your solution to the correct quiz folder. Your solution should be a .pdf file with the name <your last name>\_quiz8.pdf and uploaded. To upload your solution, please navigate to your correct Canvas ***lab*** course space. Select the “Assignments” link in the main left menu bar. Navigate to the correct quiz submission folder. Click the “Start Assignment” button. Click the “Upload File” button. Choose the appropriate .pdf file with your solution. Finally, click the “Submit Assignment” button.

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

The Four Pillars of OOP are abstraction, encapsulation, inheritance, and polymorphism.

-Abstraction:

Abstraction is the concept that only essential information is presented to the outside word, while the background details are hidden.

An example of data abstraction is rectangle, which has some important attributes like the height, the length or so on.

-Encapsulation:

It is a way of organizing or wrapping data members and methods into a structure. This provides the ability to restrict access to the data and hide information.

-Inheritance:

Inheritance allows for creating a new class from an existing class. This derived class may acquire attributes and operations of the base class. This is really useful to implement many classes with some basic attributes in common. It is considered a form of software or code reuse.

-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. Besides, it allows programs to process objects of classes that are part of the same class inheritance hierarchy as though they are part of the base class. So we can create several base-class pointers or references at compile and decide the specific object to which they point or reference at runtime.

1. **(3 pts)** Given the following function, what is the worst-case Big-O time complexity?

\_\_\_\_\_\_\_O( n3 )\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// 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;

        }

    }

}

1. **(4 pts)** In regards to inheritance, what is the *diamond* problem? Explain.

**The diamond problem occurs when a class inherits from two superclasses, and these two classes inherit from a common base class.**

**For example,**

**class People //The grandparent class**

**//The two parent classes**

**class Indian : public People**

**class American : public People**

**//The child class**

**class Indian-American : public Indian, public American**

**The Indian-American class gets two copies of all attributes of Person class, this causes ambiguities in which members are accessed.**

**When we declare an Indian-American object, constructor of Person class is called two times. And similar for destructor, when the object leaves the scope.**