**Homework 1 (Clarence D’Silva )**

**Note:** The answers to the questions not present here are in the java file comments and code.

**Question 1**

In order to estimate the running time or memory as a function of input size N, we would need to ignore the lower order terms.

For the case where N is large, the terms are almost negligible and when N is small, it does not matter.Ex 1.  ⅙ *N* 3 + 20 *N* + 16 ~   ⅙ *N* 3

2. ⅙ *N* 3 + 100 *N* 2/3  + 56 ~   ⅙ *N* 3

3. ⅙ *N* 3 + ⅙ *N* 2  + ⅙ *N* ~   ⅙ *N* 3

In the given 3 examples, the lower order terms do not have an impact to the final result and hence they can be ignored.

**Question 3**

1. Using the cost model and tilde notation to mainly simplify the counts.
2. In the case of we need to check for the number of array accesses as a function of input size N.
3. From the code in the above function "findTripleSum()" it can be noted that there are 3 array accesses in the inner loop.
4. From this in can be deduced that the array accesses are ~ 1/2 N^3. The (N^3/6) arises as N choose 3 as the number of triples

**Question 4**

The Cost model uses some basic operation as a proxy for runtime. For example, the cost model for an array access is N(N-1) assuming that the compiler or JVM does not optimize the operation.

The Tidle notation is used to estimate running time or memory as a function of input size N.

The Triple Sum uses 3 array accesses and from the Cost and Tidle model estimations it can be deduced that the array accesses is approximately 1/2N^3.

**Question5:** What are all Stack operations, explain.

Ans: The basic implementation of a stack is also called a LIFO (Last In First Out) (or) FILO (First in Last out) to demonstrate the way it accesses data, since as we will see there are various variations of stack implementations.

The three basic operations that can be performed on stacks. They are:

1. Inserting an item into a stack (push).
2. Deleting an item from the stack (pop).
3. Checking the first element of the stack also called as peek

* push, (it’s a function) insert the elements to the top of stack.
* pop, (it’s a function) which removes element from the top element of the stack.

Basic Operations Performed on Stack :-

1. Create a stack.
2. Push an element into stack
3. Pop an element from the stack
4. Check whether stack is empty or not.

**Question 7**

The Time complexity of a Stack Implemented Linked List are as follows

**Access:** O(n)

**Search:** O(n)

**Insertion:** O(1)

**Deletion:** O(1)

The Space complexity of a Stack Implemented Linked List in worst case scenario is O(n)

**Question 8**

The Time complexity of a Stack Implemented Linked List are as follows

**Access:** O(1)

**Search:** O(n)

**Insertion:** O(n)

**Deletion:** O(n)

The Space complexity of a Stack Implemented Linked List in worst case scenario is O(n)

**Push:** Every time a new element is added at the end of array. So, irrespective of number of elements already in the array, this operation will always take same time. Hence it will be O(1).

**Pop:** Every time first element of array is removed, all remaining n-1 elements are moved up. Hence it will be O(n).

**Question 9**

The Time complexity of a Stack Implemented Linked List are as follows

**Access:** O(1)

**Search:** O(n)

**Insertion:** O(n)

**Deletion:** O(n)

The Space complexity of a Stack Implemented Linked List in worst case scenario is O(n)