Session 4 Data Structure
Marcellinus Ganesia Luviandanu - 2702359572

1. A. O(n^2)
   Because there are 2 loops where both of the loops depend on n.

   B. O(n)
   Because despite having 2 loops only the outer loops that depend on n and the inner loop will only run twice.

   C. O(n^2)
   Because the loops are both dependent on the n, where the first loop will run as much as n and for the inner loops n are used to determine the value of j.

   D. O(n^2)
   Because the loops are dependant on i.

2. The content of anArray for the first loops is [0,1,2,3,3,3,3,3]
   The content of anArray for the second loops is [0,1,2,3,3,4,5,6]

3. A. Sum of an Array
   T(n) = O(1) + O(n) + O(1)
   O(n) = O(n)

   B. Matrix Multiplication
   T(n) = O(r1 x c1 x c2)
   O(n) = O(r1 x c1 x c2)

   C. For Looping
   T(n) = O(n) + O(n) + O(n) = O(3n) = O(n)
   O(n) = O(n)

   D. While Looping
   T(n) = O(log n)
   O(n) = O(1)

4. O(1) Constant Time Complexity : Average hashmap usage
   O(log n) Logarithmic Time Complexity : Binary Search
   O(n) Linear Time Complexity : Linear Search
   O(n log n) Linearithmic Time Complexity : Merge Sort
   O(n^2) Quadratic Time Complexity : Bubble Sort
   O(n^m) Polynomial Time Complexity : Matrix Multiplication
   O(2^n) Exponential Time Complexity : Recursive Fibonacci

5. ADT is a mathematical model that defines a set of data values and operations. Examples of ADTs include lists, stacks, queues, trees, graphs, sets, and maps.

6.

FEATURE	LIST	ARRAYLIST
IMPLEMENTATION	Interface	Class
MEMORY ALLOCATION	Dynamic Array	Dynamic Array
RESIZE OPERATION	Automatically resized when needed	Automatically resize when needed
INITIALIZATIOON	Cannor be directly instantiated	Can be directly instantiated
PERFORMANCE	Slower for random access and insertion	Faster for random access insertion
	Faster for sequential access and removal	Slower for sequential access and removal

7.

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<Integer> numArray = new ArrayList<>();
            numArray.add(12);
            numArray.add(25);
            numArray.add(34);
            numArray.add(46);
        System.out.println("Array before removal " + numArray);
        numArray.remove(1);
        System.out.println("Array after removal " + numArray);

    }
}
```