

1. Determine how many times the output statement is executed in each of the following fragments.

Indicate whether the algorithm is $O(n)$ or $O(n^2)$.

- a.

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        System.out.println(i + " " + j);
```
- b.

```
for (int i = 0; i < n; i++)
    for (int j = 0; j < 2; j++)
        System.out.println(i + " " + j);
```
- c.

```
for (int i = 0; i < n; i++)
    for (int j = n - 1; j >= i; j--)
        System.out.println(i + " " + j);
```
- d.

```
for (int i = 1; i < n; i++)
    for (int j = 0; j < i; j++)
        if (j % i == 0)
            System.out.println(i + " " + j);
```

- a. $O(n^2)$
- b. $O(n)$
- c. $O(n)$
- d. $O(n)$

2. Trace the execution of the following:

```
int[] anArray = {0, 1, 2, 3, 4, 5, 6, 7};
for (int i = 3; i < anArray.length - 1; i++)
    anArray[i + 1] = anArray[i];
```

and the following:

```
int[] anArray = {0, 1, 2, 3, 4, 5, 6, 7};
for (int i = anArray.length - 1; i > 3; i--)
    anArray[i] = anArray[i - 1];
```

What are the contents of `anArray` after the execution of each loop?

- a. [0, 1, 2, 3, 3, 3, 3, 3]
- b. [0, 1, 2, 3, 3, 4, 5, 6]

3. Please provide analysis to calculate $O(n)$ and $T(n)$ for the following algorithms:

a. Sum of an Array

```
public static int sumArray(int[] array) {
    int sum = 0; // 1 operation
    for(int i = 0; i < array.length; i++) { // n iterations
        sum += array[i]; // 2 operations (access and addition)
    }
    return sum; // 1 operation
}
```

b. Matrix Multiplication

```
public static int[][] multiplyMatrices(int[][] firstMatrix, int[][] secondMatrix,
int r1, int c1, int c2) {
    int[][] product = new int[r1][c2];
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++) {
            for (int k = 0; k < c1; k++) {
                product[i][j] += firstMatrix[i][k] * secondMatrix[k][j];
            }
        }
    }
    return product;
}
```

a. $T(n) = 2n + 2$

b. $T(n) = n^3 + 2$

Please provide analysis to calculate $O(n)$ and $T(n)$ for the following algorithms:

c. For Looping

```
int result = 0;
for (int i = 0; i < n; i++) {
    result = i + i;
}
for (int j = 0; j < n; j++) {
    result = j + j;
}
for (int k = 0; k < n; k++) {
    result = k + k;
}
```

d. While Looping

```
int i = n;
while (i > 0) {
    int k = 2 * i;
    i = i / 2;
}
```

c. $T(n) = 3n$

d. "I dont understand"

4.

Constant Time Complexity - $O(1)$

Linear Time Complexity - $O(n)$

Logarithmic Time Complexity - $O(\log n)$

Quadratic Time Complexity - $O(n^2)$

Exponent - $O(2^n)$

5.

ADT defines a set of operations on the data and their behavior. It provides a logical description of the data and operations.

Examples - Stack, Queue, Set

6.

List	ArrayList
Slower for random access, yet efficient for adding/removing elements in the middle index	Faster for random access
Can have different implementations such as arraylist and linkedlist	Specifically uses an array to store elements

7.