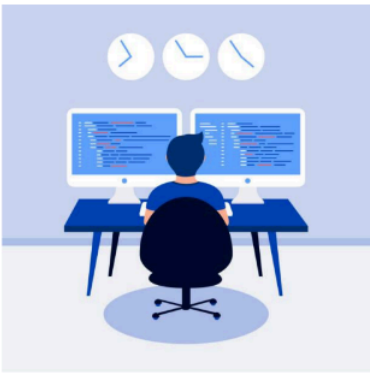
Week 3: In class assignment and Project task

Assignment 1:

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) The notation for this code snippet is O(n^2), it grows quadratically because there are two nested inputs that go from o to n, where n is determined by the input and not constant.
b) The notation for this code snippet is O(1), there are two nested loops although one of the loops is determined by n input the other has a fixed value making the growth constant.
c) The notation for this code snippet is O(1), the nested loops cause the loop to go on forever because value j will always be smaller than i in the second loop but the growth will still only be constant and the input doesn't determine the growth.
d) The notation for this code snippet is O(1), the nested second loops will always return a 0 because value j % i will not be equal to 0.

Assignment 2:

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, 5, 6, 7]
[0, 1, 2, 3, 3, 3, 6, 7]
[0, 1, 2, 3, 3, 3, 3, 7]
[0, 1, 2, 3, 3, 3, 3, 3]

b)
[0, 1, 2, 3, 4, 5, 6, 6]
[0, 1, 2, 3, 4, 5, 5, 6]
[0, 1, 2, 3, 4, 4, 5, 6]
[0, 1, 2, 3, 3, 4, 5, 6]

Assignment 3:

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) The time complexity for this is O(n) linear, the sum grows linearly depending on the elements of the array.
   The space complexity for this is O(1), the space used by the code is linear and does not depend on input so it is constant.
b) The time complexity for this is $O(n^2)$, the product is determined by the input.
   The space complexity for this is

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) The time complexity for this is O(n), it is directly proportional to the n so its linear
The space complexity for this is O(1), only the variable **result** is used, so its a constant space complexity

d) The time complexity for this is O (log n),
The space complexity for this O(1), only i and k variables are used

Assignment 4:
Examples of time complexity in popular algorithms:
Constant Time Complexity (O(1)): Pushing or popping the top of a stack
Linear Time Complexity (O(n)): Counting the occurrence of a specific element in an array
Quadratic Time Complexity (O(n^2)): Bubble sort
Exponential Time Complexity (O(2^n)): Computing a combination of a set

Assignment 5:
Abstract data structures are models of how data can be organised and manipulated. They help us with data organisations and enable efficient algorithms.

Assignment 6:

Arraylist	List
Allows fast access and retrieval of elements using an index.	Slower for random access but efficient for adding/removing in the middle index
Represents a resizable array.	Represents a generic interface for lists.
Provides common operations for lists, such as adding, removing, and accessing elements.	Specifies common operations for lists, including adding, removing, and accessing elements.

Assignment 7:

```
import java.util.ArrayList;

public class Main {
```

```
    public static void main(String[] args) {
        // Create an ArrayList to store integers
        ArrayList<Integer> numbers = new ArrayList<>();

        // Add elements to the ArrayList
        numbers.add(12);
        numbers.add(25);
        numbers.add(34);
        numbers.add(46);

        // Remove number 25 from the ArrayList
        numbers.remove(Integer.valueOf(25));

        // Print the ArrayList
        System.out.println(numbers);
    }
}
```

Result:

```
[12, 34, 46]

Process finished with exit code 0
```