CCC2113 Data Structures and Algorithms Analysis Name:

Lab 2: Algorithm Complexity (18/4/2024) Group:

Question 1: Analyze the time complexity of the following code segments and write it in Big O notation:

```
1  static void printArray(int n) {                                    1
2      java.util.Random random = new java.util.Random();               1
3      int[] numbers = new int[n];                                     1
6      for (int i = 0; i < n; i++) {                                   n+1
7          numbers[i] = random.nextInt(100);                           n
8      }
11     System.out.println("Random numbers in the array:");             1
12     for (int i = 0; i < numbers.length; i++) {                      n+1
13         System.out.println(numbers[i]);                             n
14     }
15 }
```

Total Frequency: 4n+6

Complexity in Big O: O(n)

Question 2: Find the frequency count of each statement in the following code segments:

2.1)

```
1. public class SearchElement {                                        1
2.     public static void main(String[] args) {                        1
3.         int[] arr = {10, 20, 30, 40, 50, 60, 70, 80, 90};           1
4.         int target = 10;                                            1
5.         boolean found = false;                                      1
6.
7.         for (int i = 0; i < arr.length; i++) {                      n+1
8.             if (arr[i] == target) {                                 n
9.                 System.out.println("Element found at index: " + i); 1
10.                found = true;                                       1
11.                break; // Exit the loop once the element is found   1
12.            }
13.        }
14.        if (!found) {                                               1
15.            System.out.println("Element not found in the array."); 1
16.        }
17.    }
18. }
```

Total Frequency: _____2n+11____

2.2) Find the frequency count of the above code segment with `target = 90;`

2(9)+11= 29

2.3) Write the complexity of the above code segment using **asymptotic notations** when `target = 10` and when `target = 90`

$target = 10 : \omega(1)$ $target = 90 : O(9)$

Analyze the time complexity of the following code segments and write it in Big O notation:

```
1. public class Sort {                                                    1
2.     public static void Sort(int[] arr) {                               1
3.         int n = arr.length;                                            1
4.         for (int i = 1; i < n; ++i) {                                  n
5.             int key = arr[i];                                          n-1
6.             int j = i - 1;                                             n-1
7.
8.             /* Move elements of arr[0..i-1], that are greater than key,
9.                to one position ahead of their current position */
10.            while (j >= 0 && arr[j] > key) {                           n(n-1)
11.                arr[j + 1] = arr[j];                                   n(n-1)
12.                j = j - 1;                                             n(n-1)
13.            }
14.            arr[j + 1] = key;                                          n-1
15.        }
16.    }
17.
18.    public static void main(String[] args) {                           1
19.        int[] arr = {12, 11, 13, 5, 6};                                1
20.
21.        System.out.println("Original array:");                         1
22.        printArray(arr);                                               1
23.
24.        Sort(arr);
25.
26.        System.out.println("\nSorted array:");                         1
27.        printArray(arr);                                               1
28.    }
29.
30.    private static void printArray(int[] arr) {                        1
31.        for (int value : arr) {                                        n
32.            System.out.print(value + " ");                             n
33.        }
34.        System.out.println();                                          1
35.    }
36. }
```

Complexity: ___Big O(n^2)