CCC2113 Data Structures and Algorithms Analysis **Name**:

Lab 6: Sort & Search Algorithms (23/5/2024) **Group**:

1. The following are the algorithm steps of Selection Sort:

```
1. Begin with the first element in the array. This will be your current
   position.
2. Compare with all the elements from your current position to the end of
   the array.
3. Identify the smallest element.
4. If the smallest element is not already the current position, swap it
   with the element at your current position.
5. Consider the next element as your new current position and find the
   smallest element from here to the end of the array.
6. Continue finding the smallest element and swapping until you reach the
   second-to-last position in the array. By then, the array should be
   sorted.
```

No, write a method `selectionSort` in java that takes an array of integers and sorts it in ascending order using the selection sort algorithm.

```java
public class SelectionSort {
    public static void selectionSort(int[] array) {
        int n = array.length;

        // Traverse through all array elements
        for (int i = 0; i < n - 1; i++) {
            // Find the minimum element in unsorted array
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (array[j] < array[minIndex]) {
                    minIndex = j;
                }
            }

            // Swap the found minimum element with the first element
            if (minIndex != i) {
                int temp = array[minIndex];
                array[minIndex] = array[i];
                array[i] = temp;
            }
        }
    }

    public static void main(String[] args) {
        int[] array = {64, 25, 12, 22, 11};
        selectionSort(array);
        System.out.println("Sorted array:");
        for (int i = 0; i < array.length; i++) {
            System.out.print(array[i] + " ");
        }
    }
}
```

2. Write a method `findThird` that takes an array of integers with size more than 3 and returns the 3rd smallest element in the array by sorting the array using the selection sort algorithm.

```java
public class FindThirdSmallest {
    // Method to sort the array using selection sort algorithm
    public static void selectionSort(int[] array) {
        int n = array.length;

        // Traverse through all array elements
        for (int i = 0; i < n - 1; i++) {
            // Find the minimum element in unsorted array
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if (array[j] < array[minIndex]) {
                    minIndex = j;
                }
            }

            // Swap the found minimum element with the first element
            if (minIndex != i) {
                int temp = array[minIndex];
                array[minIndex] = array[i];
                array[i] = temp;
            }
        }
    }

    // Method to find the 3rd smallest element in the array
    public static int findThird(int[] array) {
        // Check if array length is more than 3
        if (array.length < 3) {
            throw new IllegalArgumentException("Array size must be greater than 3");
        }

        // Sort the array using selection sort
        selectionSort(array);

        // Return the 3rd smallest element
        return array[2];
    }

    public static void main(String[] args) {
        int[] array = {64, 25, 12, 22, 11, 3, 5};
        try {
            int thirdSmallest = findThird(array);
            System.out.println("The 3rd smallest element is: " + thirdSmallest);
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }
    }
}

```

3. Write a method `findAll` that takes an array of integers and a target integer. The method should print a list of all indices where the target is found in the array using Binary Search algorithm]

Example:

Input: array = [1, 2, 3, 2, 4, 2, 5] and target = 2 [3 marks]

Output: *1*

3

5

```
public static void findAll(int[] array, int target) {
    // Sort the array using selection sort
    selectionSort(array);

    // Find one occurrence of the target using binary search
    int index = binarySearch(array, target);

    if (index == -1) {
        System.out.println("Target not found");
        return;
    }

    // List to store all indices of the target
    ArrayList<Integer> indices = new ArrayList<>();

    // Expand outwards to find all occurrences
    // Add the found index
    indices.add(index);

    // Check left side of the found index
    int leftIndex = index - 1;
    while (leftIndex >= 0 && array[leftIndex] == target) {
        indices.add(leftIndex);
        leftIndex--;
    }

    // Check right side of the found index
    int rightIndex = index + 1;
    while (rightIndex < array.length && array[rightIndex] == target) {
        indices.add(rightIndex);
        rightIndex++;
    }

    // Print all indices
    indices.sort(null); // Sort indices to ensure they are in ascending order
    for (int idx : indices) {
        System.out.println(idx);
    }
}

```