Objective: more on statements and control flow, functional abstraction, a little bit on arrays.

Assignment: The goal of this assignment is to implement the quicksort algorithm according to the pseudocode given below.

Let arr denote an array of integers. In Java, arr[i] refers to the value at the i'th position of the array arr. For example, arr[0] denotes the value at the first position of arr and arr[11] denotes the value at the 11'th position of arr.

Quicksort sorts an array of integers arr by first partitioning the array into two subarrays arr1 and arr2 where arr1 contains all integers less than or equal to a pivot value x and arr2 contains all integers greater than x. After partitioning, Quicksort proceeds to sort the two subarrays arr1 and arr2 recursively. The pseudocode for quicksort is given below.

```java
// sorts the elements in the array arr from position left to position right
procedure quicksort(int[] arr, int p, int r)
    if (left < right) {
        q = partition(arr, p, r);
        quicksort(arr, p, q-1);
        quicksort(arr, q+1, r);
    }

// returns an integer q such that
// every element in positions left to q-1 of arr is less than or equal
// to x
// every element in positions q+1 to right of arr is greater than x
procedure partition(int[] a, int left, int right)
    let x = arr[left];
    let i = left - 1;
    let j = right + 1;

    repeat
        increment i until a[i] is an element greater than x
        decrement j until a[j] is an element less than or equal to x
        if (i<j) then
            swap arr[i] with a[j]
        until (i>=j)
    swap arr[left] with a[j]
```

1
You should implement quicksort by completing the following Java code. Note that you should handle boundary conditions to avoid ArrayOutOfBoundsException in Java. For example, if your array arr is of size 5, then accessing arr[0], ..., arr[4] is OK. However, if you attempt to access arr[-1] or arr[5], Java will give you an ArrayOutOfBoundsException.

```java
import tio.*;

class QuickSort {
    static void quicksort(int[] arr, int p, int r) {
        // fill in code
    }

    static int partition(int[] arr, int p, int r) {
        // fill in code
    }

    public static void main(String[] args) {
        System.out.println("Enter total number of elements to sort:");
        int i = Console.in.readInt(); // the size of the array
        int[] numbers = new int[i]; // construct an integer array of size i
        for (int j=0; j<i; j++) {
            System.out.print("Please enter number "+j+": ");
            numbers[j] = Console.in.readInt();
        }

        System.out.print("The list of numbers is: ");
        for (int j=0; j<i; j++) {
            System.out.print(numbers[j] + " ");
        }
        System.out.println();
        quicksort(numbers, 0, i-1);

        System.out.println("The sorted result is:");
        for (int j=0; j<i; j++) {
            System.out.print(numbers[j] + " ");
        }
        System.out.println();
    }
}
```

Some sample runs of quicksort are given below:

```
my_prompt> java QuickSort
Enter total number of elements to sort:
1
Please enter number 0: 28
```
The list of numbers is: 28
The sorted result is:
28
my_prompt> java QuickSort
Enter total number of elements to sort: 
2
Please enter number 0: 23
Please enter number 1: 20
The list of numbers is: 23 20
The sorted result is:
20 23
my_prompt> java QuickSort
Enter total number of elements to sort: 
5
Please enter number 0: 34
Please enter number 1: 1
Please enter number 2: 88
Please enter number 3: 88
Please enter number 4: 1
The list of numbers is: 34 1 88 88 1
The sorted result is:
1 1 34 88 88
my_prompt>

Programming Assignment 3 Submission

- Submit your source code (the .java file only) through your WebCT account.
- Submit early & submit often. You can submit your code as many times as you want until the deadline, so you may want to submit well before the deadline, even if you only have a partial solution.