CS526 O2 – Spring 2020
Homework Assignment 2

This assignment has five parts.

Part 1 (20 points). Write a Java program named *Hw2_part1.java*. This program must include a recursive method named *evenCount* that receives an array of integers *a* and returns the number of even integers in *a*. If the given array is:

```
int[] a  = {35, 12, 57, 28, 49, 100, 61, 73, 92, 27, 39, 83, 52};
```

Then, the *evenCount* method must return 5 and the program output must be:

```
Number of even integers = 5
```

The signature of your method must be:

```
evenCount(int[] a)
```

Note that the signature is incomplete. You are responsible for writing a complete signature.

You may want to consider writing another method with additional parameters (refer to page 214 of the textbook), which is invoked by the *evenCount* method. In this case, the *evenCount* method itself is not a recursive method and the method with additional parameters will be a recursive method.

You also need to write a *main* method to test the above method(s).

Part 2 (20 points). Write a Java program named *Hw2_part2.java*. This program must include a recursive method named *lessThanKFirst* that receives an array of integers *a* and an integer *k* and rearranges the integers in *a* in such a way that all integers that are smaller than *k* come before any integers that are greater than or equal to *k*. As an example, suppose that *a* and *k* are:

```
int[] a  = {35, 12, 57, 28, 49, 100, 61, 73, 92, 27, 39, 83, 52};
int k = 73;
```

Then, the following is one possible output:

```
a = [35, 12, 57, 28, 49, 52, 61, 39, 27, 92, 83, 73, 100]
```

Your output may be different from the above output. As long as all integers smaller than 73 come before those that are greater than or equal to 73, that is OK.

You must not use another array when you rearrange the elements of the given array and you must not sort the given array (a sorted array automatically satisfies the given requirement).

The signature of your method must be:

```
lessThanKFirst(int[] a, k)
```

Note that the signature is incomplete. You are responsible for writing a complete signature.

You may want to consider writing another recursive method with additional parameters (refer to page 214 of the textbook), which is invoked by the *lessThanKFirst* method. In this case, the *lessThanKFirst* method itself is not a recursive method and the method with additional parameters will be a recursive method.

You also need to write a *main* method to test the above method(s).

Part 3 (30 points). This is a practice of analyzing the running time of small programs. Express the running time of each of the following Java methods using the *big-oh* notation. You must justify your answer. Otherwise, you will lose points even if your answers are correct.

```
public static int method1(int[] arr) {
  int n = arr.length, total = 0;
  for (int j=0; j < n; j++)
    total += arr[j];
  return total;
}
```

```
public static int method2(int[] arr) {
  int n = arr.length, total = 0;
  for (int j=0; j < n; j++)
    for (int k=0; k <= j; k++)
      total += arr[j];
  return total;
}
```

```
// initial invocation: method3(a, 0, a.length-1)
public static int method3(int[] a, int x, int y) {
  if (x >= y) return a[x];
  else {
    int z = (x + y) / 2;
    int u = method3(a, x, z);
    int v = method3(a, z+1, y);
    if (u < v) return u;
    else return v;
  }
}
```

```
public static int method4(int n) {
    if (n < 1) {return 1;}
    else {return (n + method4(n / 2));}
}
```

Part 4 (10 points). Consider the following recursive method:

```
public static int method5(int x, int y) {
    if (y == 0)  return 1;
    else {
        int w = method5(x, y/2);
        int z = 2*w;
        if (y % 2 == 0)
            z *= x;
        System.out.println("z = " + z);
        return z;
    }
  }
```

If this method is invoked with *method*5(5, 10), it will go through a sequence of recursive calls. Show the return value of each recursive call in the order the results are generated.

Part 5 (20 points). This part is a practice of stack operations and queue operations.

Part 5-1. Apply the following operations, one at a time in the given order, on an initially empty stack, and show the stack content and the output after each operation. If there is no output, then write *none*.

Operation	Stack content	Output
push(5)		
push(3)		
pop( )		
push(2)		
size( )		
pop( )		
push(8)		
pop( )		
top( )		
size( )		
isEmpty( )		

Part 5-2. Apply the following operations, one at a time in the given order, on an initially empty queue, and show the queue content and the output after each operation. If there is no output, then write *none*.

Operation	Queue content	Output
enqueue(5)		
enqueue(3)		
dequeue( )		
enqueue(2)		
size( )		
dequeue( )		
enqueue(8)		
dequeue( )		
first( )		
size( )		
isEmpty( )		

Documentation

No separate documentation is needed. However, you must include sufficient inline comments within your program. Your inline comments must include a specification for each method in your program. A specification of a method must include at least the following:

- Brief description of what the method does
- Input parameters: Brief description of parameters and their names and types, or none
- Output: Brief description and the type of the return value of the method, or none

If you want, you can write a specification using "precondition" and "postcondition" but this is not required.

Deliverables

You must submit three files – *Hw2_part1.java*, *Hw2_part2.java*, and *Hw2_others.pdf*. The *Hw2_others.pdf* file must include answers to parts 3, 4, and 5. Combine the three files (and other additional files, if any) into a single archive file, name it *LastName_FirstName_hw2.EXT*, where *EXT* is an appropriate file extension, such as *zip* of *rar*, and upload it to Blackboard.

Grading

Part 1: Your program will be tested with two arrays and up to 6 points will be deducted for each wrong output.

Part 2: Your program will be tested with two pairs of (a, k) and up to 6 points will be deducted for each wrong output.

Part 3: 5 points will be deducted for each wrong answer.

Part 4: Up to 6 points will be deducted if your answer is wrong.

Part 5: 1 point will be deducted for each wrong answer (stack/queue content or output) up to 12 points.

Up to 20 points will be deducted if your program does not have sufficient inline comments.