CMPS 12B
Introduction to Data Structures
Programming Assignment 1

The purpose of this assignment is to gain experience implementing recursive algorithms in Java. You will write five recursive methods with headings

```
static void reverseArray1(int[] X, int n, int[] Y)
static void reverseArray2(int[] X, int n, int[] Y)
static void reverseArray3(int[] X, int i, int j)
static int maxArrayIndex(int[] X, int p, int r)
static int minArrayIndex(int[] X, int p, int r)
```

Functions `reverseArray1()` and `reverseArray2()` are similar to the functions of the same names in the example `Reverse.java` posted on the class webpage

https://classes.soe.ucsc.edu/cmps012b/Winter18/Examples/Lecture/Recursion/Reverse.java

which were discussed in class. These two functions will copy the elements in the input array `X[]` into the output array `Y[]` in reverse order (instead of printing them to `stdout`, like the examples in class.)

Function `reverseArray1()` will copy the leftmost n elements in `X[]` into the rightmost n positions in `Y[]` in reverse order. It will do this by first copying the n^{th} element from the left in `X[]` into the n^{th} position from the right in `Y[]`, then calling itself recursively to place the leftmost $n-1$ elements in `X[]` into the rightmost $n-1$ positions in `Y[]` in reverse order. The recursion 'bottoms out' when the length of the subarray being reversed is zero, in which case the function returns without doing anything. To reverse the entire array, one calls `reverseArray1()` with n equal to the length of the input array `X[]`. It is assumed that the calling function has allocated the output array `Y[]` before the function is called.

Function `reverseArray2()` performs a 'dual' procedure. Its description can be obtained by interchanging the words 'left' and 'right' everywhere you see them in the preceding paragraph.

Function `reverseArray3()` is very different from the first two functions in that it actually alters the order of the elements in `X[]`, rather than just copying them in reverse order into another array. This function reverses the subarray `X[i,..,j]` consisting of those elements from index i to index j, inclusive. It does this by first swapping the elements at positions i and j, then calling itself recursively on the subarray `X[i+1,...,j-1]` from index $i+1$ to index $j-1$. To reverse the entire array, one calls `reverseArray3()` with i equal to 0 and j equal to `X.length-1`.

Function `maxArrayIndex()` returns the index of the maximum element in the subarray `X[p,..,r]`, and function `minArrayIndex()` returns the index of the minimum element in the same subarray. These follow a procedure similar to that of function `mergeSort()` which can be found on the webpage at

https://classes.soe.ucsc.edu/cmps012b/Winter18/Examples/Lecture/Recursion/MergeSort.java

and which will be discussed in class. Function `maxArrayIndex()` operates as follows. If subarray `X[p...r]` contains just one element, its index is returned. If subarray `X[p...r]` contains more than one element, the middle index is computed as `q=(p+r)/2`, the indices of the maximum elements in subarrays `A[p...q]` and `A[q+1...r]` are computed recursively, then the index of the larger of the two maxima is

returned. Function `minArrayIndex()` follows a similar procedure to return the index of the smallest value in the subarray `A[p...r]`.

All these methods will be inserted into a class called `Recursion`, and will be called from function `main()` defined below.

```
public static void main(String[] args){

   int[] A = {-1, 2, 6, 12, 9, 2, -5, -2, 8, 5, 7};
   int[] B = new int[A.length];
   int[] C = new int[A.length];
   int minIndex = minArrayIndex(A, 0, A.length-1);
   int maxIndex = maxArrayIndex(A, 0, A.length-1);

   for(int x: A) System.out.print(x+" ");
   System.out.println();

   System.out.println( "minIndex = " + minIndex );
   System.out.println( "maxIndex = " + maxIndex );

   reverseArray1(A, A.length, B);
   for(int x: B) System.out.print(x+" ");
   System.out.println();

   reverseArray2(A, A.length, C);
   for(int x: C) System.out.print(x+" ");
   System.out.println();

   reverseArray3(A, 0, A.length-1);
   for(int x: A) System.out.print(x+" ");
   System.out.println();

}
```

The output of this program will be the six lines

```
-1 2 6 12 9 2 -5 -2 8 5 7
minIndex = 6
maxIndex = 3
7 5 8 -2 -5 2 9 12 6 2 -1
7 5 8 -2 -5 2 9 12 6 2 -1
7 5 8 -2 -5 2 9 12 6 2 -1
```

Place your functions, along with function `main()` above in the class `Recursion`, and place that class definition in a file called `Recursion.java`. A template for this file will be placed on the examples section of the webpage. As you write your functions, test them on *many* different input arrays to thoroughly check their operation. When you submit the project though, include function `main()` exactly as written above.

Write a `Makefile` that creates an executable jar file called `Recursion`. Include a clean utility that removes all `.class` files and the executable jar file from the current directory. (See lab1 to understand how to do all this.) Submit the files `README`, `Makefile`, and `Recursion.java` to the assignment name `pa1` by the due date.