COP 3503C Computer Science II Section 2 – Fall 2019

ASSIGNMENT 1

1. Please write a Divide-and-Conquer Java algorithm solving the following problem:

Given an "almost sorted" array of **distinct** integers, and an integer x, return the index of x in the array. If the element x is not present in the array, return -1.

"Almost sorted" means the following. Assume you had a sorted array A[0…N], and then split it into two pieces A[0…M] and A[M+1…N], and move the second piece upfront to get the following:

A[M+1]…A[N]A[0]…A[M].

Thus, the "almost sorted" array is either a sorted array, or it consists of two sorted subarrays, such that every element of the first subarray is greater or equal than every element of the second subarray.

For example, the array

{3, 17, 28, 935, 1011, **-10, 0, 2**}

is "almost sorted" since it consists of two sorted subarrays: {3, 17, 28, 935, 1011} and {-10, 0, 2} with the property that each element in the first subarray is greater or equal than every element of the second subarray.

Note: One of the subarrays can be empty, i.e., the array might be sorted.

You need to develop an efficient modification of the Binary Search Algorithm, with worst-case running time of $O(\lg n)$ for an array of n elements.

Reminder: In Java, elements of an array of n elements have indexes 0…n-1.

Formally speaking, your input is an array of distinct integers, and the element x to find; your output is: the index of x in the array, or -1 in case x is not there.

With the array above and x=935, the algorithm has to return 3 (the index of the element 935 in the array).

Please develop the following Java function:

```
public static int FindIndex(int[] arr, int x)
```

Here **arr** is the array of distinct integers, **x** is the element to find.

NOTE: In your algorithm, you do not have to check that the array is "almost sorted". However, you have to check "boundary cases" like an empty array.

2. For Problem 1, please estimate the worst-case Running Time for a random array of size n, and prove that it is indeed $O(\lg n)$. Please show all your work. Just stating something like "since

Binary Search runs in $\boldsymbol{O}(\lg \boldsymbol{n})$ time, our algorithm has the same runtime estimate" is not enough. A rigorous explanation is expected.

3. **The Power Set** of a set of (**distinct)** objects **S** is the set of all subsets of **S**, including **S** itself and an empty set. For example, let **S** be a set of words: **S**={Hello, World, Students}. Then possible subsets are
{Hello}
{World}
{Students}
{Hello, World}
{Hello, Students}
{World, Students}
{Hello, World, Students}
∅
From a course in Discrete Mathematics, you should know that if **S** contains *n* elements, then its power set consists of 2^n elements.

Please write a recursive algorithm, which will take an array of objects as an input and return an array of arrays of objects as the power set.

In the previous example, our objects were strings, and the output was an array of arrays of strings. Right?
In order to do that, **please develop the following method**

```
public static Object[][] GeneratePowerSet(Object[] arr)
```

Please do not forget to check "boundary cases" and possible invalid inputs like some objects in arr are null. If that happens, your function should print an error message and return null. (Note that in real world programming, I would suggest to throw an exception instead of printing an error message; we will talk about this later in our course.)
Please also test the method by feeding it an array of integers and printing out the power set. You can do this for testing:

```
Integer[] my_arr = {-2,7,10,23,19};
Integer[][] pSet = GeneratePowerSet(my_arr);

for(int i=0; i<pSet.length; i++) {
      System.out.print("{");
      for(int j=0; j<pSet[i].length; j++) {
            System.out.print(pSet[i][j]);
            if(j+1 < pSet[i].length)
                  System.out.print(", ");
      }
      System.out.println("}");
}
```

Side Note: Please note that if you want to print out an object (whatever it means) by calling
`System.out.println(my_object);`
the toString() method of that object would be called inside println(). It is the same as the following call:
`System.out.println(my_object.toString());`

I encourage you to exercise with toString() method for a simple class ComplexNumber you can easily create:

```
class ComplexNumber {
	public double Re;
	public double Im;
	public ComplexNumber(double x, double y) {
		super();
		this.Re = x;
		this.Im = y;
	}
	public String toString() {
		// your code inside returning something like (Re,Im)
	}
}
```

I am going to publish test arrays and corresponding output for Problem 1 and Problem 3 later. So, you could test your solutions. However, I would strongly encourage you to prepare your own test arrays (think about special cases!). This is a very important skill of a Software Engineer.