Lab session 9 (Recursion 2)
CSC 113
King Saud University
Computer and Information Sciences

1 String manipulation

Write the `class ArrayRecursor`. For each recursive method, write a `public` method to provide a clean interface, and a `private` helper method to perform the recursion. You should think carefully about what the helper method needs to have as parameters.

- Do **not** use loops.
- Do **not** give the class any attributes
- Do **not** use static variables.

1.1 Exercise 1

Write the `static`, recursive method `reverseArray` which receives an array of integers and reverses it *in place*. The method does not return anything.

- Note: Do not create a new string.

```
1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 2
[1,2,3,4]
1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 3
Array reversed!
```

```
1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 2
[4,3,2,1]
```

1.2 Exercise 2

Write the `static`, recursive method `occurances` which receives an array a of integers, an integer x and returns the number of times that x appears in a.

```
1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 4
Enter a number: 2
The number 2 occurs 3 times in [1,2,5,0,2,3,2]
```

1.3 Exercise 3

Write the `static`, recursive method `palindrome` which receives a string s and returns true if s is a palindrome and returns false otherwise.

```
1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 5
The array [1,2,3,4] is not a palindrome
```

```
1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 5
The array [1,2,3,2,1] is a palindrome
```

1.4 Exercise 6

Write the `static`, recursive method `isSorted` which receives an integer array a and returns true if a is in increasing order, and false otherwise.

1.5 Exercise 5

Write the `static`, recursive method `mergeTwo` which receives two arrays of *sorted* integers and returns a merged, *sorted* array containing the elements of both. Your method should verify that both arrays are sorted first. If either of two arrays is not sorted, return an empty array. In the running example, we merge `[1,2,5,7,9,12]` with `[4,8,9,10,42]`

```
1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 2
[1,2,5,7,9,12]

1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 6
How large is the array? 5
Enter the array: 4 8 9 10 42
Merged: [1,2,4,5,7,8,9,9,10,12,42]
```

1.6 Exercise 6

Write the `static`, recursive method `binarySearch` which receives an array of *sorted* integers and an integer x. If x is in the array, the method returns the index of x. Otherwise, it returns -1. *You may assume that the array is sorted.* Binary search works as shown in the example. If looking for the number 7 in the array $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$:

```
Step 1: [1,2,3,4,5,6,7,8,9,10]
                ^---------------- Is this 7? No! 7 must be in [6,7,8,9]
Step 2: [1,2,3,4,5,6,7,8,9,10]
                      ^---------- Is this 7? No! 7 must be in [6,7]
Step 3: [1,2,3,4,5,6,7,8,9,10]
                    ^------------ Is this 7? No! 7 must be in [7]
Step 4: [1,2,3,4,5,6,7,8,9,10]
                      ^---------- Is this 7? Yes!
```

Note that we found 7 in four steps. An ordinary search would take 7 steps. Another example: Finding 3 in $[1, 2, 4, 5, 10, 21, 23, 24, 26]$

```
Step 1: [1,2,4,5,10,21,23,24,26]
```

```
                  ^------------------ Is this 7? No. 7 must be in [1,2,4,5]
Step 2: [1,2,4,5,10,21,23,24,26]
            ^---------------------- Is this 7? No. 7 must be in [4,5]
Step 3: [1,2,4,5,10,21,23,24,26]
              ^-------------------- Is this 7? No. 7 must be in [5]
Step 4: [1,2,4,5,10,21,23,24,26]
                ^------------------ Is this 7? No. 7 is not in the array!
```

Your method should receive an array, an integer and return the index if found, -1 if not found. Example output of a program using `binarySearch`.

```
1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 2
[1,2,3,4,5,6,7,8,9,10]

1) Enter a new array.
2) Print current array.
3) Reverse current array.
4) Count occurances.
5) Check if array is palindrome
6) Merge with another sorted array.
7) Search in the sorted array.
8) Quit
Enter a choice: 7
7 was found at index 6
```

1.7 Exercise 7

Write a main program which provides the menu and prompts the user to choose from 1 to 8. Offer one choice for each recursive method and test them. Your program should quit when the user chooses to quit.