COMP 53: Algorithm Analysis, part 2

Instructions: In this lab, we are going to review recursion.

- Get into groups of **at most two people** to accomplish this lab.
- At the top of your source code files list the group members as a comment.
- Each member of the group must individually submit the lab in Canvas.
- This lab includes **26 points** in aggregate. The details are given in the following.

1 `main.cpp`

In `main.cpp` do the following step by step:

1. Globally define array `a[]` of integers consisting of values: 2,8,1,7,3 in order.
2. Globally define array `b[]` of integers, without initial values.
3. Define recursive function `void printArray(int a[], int size, int index)` that receives an array of integers along with its size, and index (an integer) of the current element. It prints the elements in the standard output. Do not use any loops, and rely on recursion to traverse the array ***(2 points)***.
4. Define recursive function `void printReverseArray(int a[], int size, int index)` that receives an array of integers along with its size, and index (an integer) of the current element. It prints the elements in the standard output in the reverse format. Do not use any loops, and rely on recursion to traverse the array ***(2 points)***.
5. Define recursive function `void reverseArray(int a[], int start, int end)` that receives an array of integers along with an start index, and and end index. It reverses the elements in the input array that are indexed between `start` and `end`. Do not use any loops, and rely on recursion to traverse the array ***(2 points)***.
6. Define recursive function `int sumArray(int a[], int size, int index)` that receives an array of integers along with its size, and index (an integer) of the current element. It returns the summation of all numbers in the array. Do not use any loops, and rely on recursion to traverse the array ***(2 points)***.
7. Define recursive function `int productArray(int a[], int size, int index)` that receives an array of integers along with its size, and index (an integer) of the current element. It returns the production of all numbers in the array. Do not use any loops, and rely on recursion to traverse the array ***(2 points)***.
8. Define recursive function `void copyArray(int a[], int b[], int size, int index)` that receives two integer arrays, the size of the first array, and an index (an integer) of the current element. It copies the first array into the second array. Do not use any loops, and rely on recursion to traverse the array ***(2 points)***.
9. Define recursive function `int minArray(int a[], int size, int index)` that receives an array of integers along with its size, and index (an integer) of the current element. It returns the minimum of all numbers in the array. Do not use any loops, and rely on recursion to traverse the array ***(2 points)***.
10. Define recursive function `int maxArray(int a[], int size, int index)` that receives an array of integers along with its size, and index (an integer) of the current element. It returns the maximum of all numbers in the array. Do not use any loops, and rely on recursion to traverse the array ***(2 points)***.

In `main()` function do the following step by step, using the functions defined above:

(a) Print out the contents of array `a[]` in order ***(1 points)***.

(b) Print out the contents of array `a[]` in reverse (do not reverse the array) ***(1 points)***.

(c) Print out the contents of array `a[]` after reversing it ***(1 points)***.

(d) Print out the summation of elements of array `a[]` ***(1 points)***.

(e) Print out the product of elements of array `a[]` ***(1 points)***.

(f) Copy array `a[]` to `b[]` and print out `b[]`'s elements ***(1 points)***.

(g) Print out the minimum element of array `a[]` ***(1 points)***.

(h) Print out the maximum element of array `a[]` ***(1 points)***.

(i) All of the recursive functions have the similar recurrence runtime relation. Comment the recurrence relation in `main()` in terms of big-O ***(1 points)***.

(j) What is the runtime complexity according to the recurrence relation? Comment it in `main()` ***(1 points)***.

The output of the program may look like the following:

```
Array a[] content in order:
2 8 1 7 3

Array a[] content in reverse:
3 7 1 8 2

Reversing a[]:
3 7 1 8 2

Summation of elements in a[]:
21

Product of elements in a[]:
336

Copying a[] to b[]:
3 7 1 8 2

Minimum element in a[]:
1

Maximum element in a[]:
8
```