Timothy\_Zwart\_CMPT\_435

Assignment 7

**Date Assigned: 10/26/2018**

**Due: Midnight 11/02/2018 on iLearn**

**Please read turn-in checklist at the end of this document before you start doing exercises.**

**Section 1: Pen-and-paper Exercises**

1. Input: A list of alphabetic characters.

Output: Write a recursive algorithm that determines whether or not the string “marist” occurs in X, and if so, to return the location of the rightmost (or say, the last) occurrence of “marist”.

For example, given {'a', 'm', 'a', 'r', 'i', 's', 't', 'm', 'a', 'r', 'i', 's', 't', 'v', 'h', 'e'}, output: 7. Given {'o', 't', 'z', 'y', 'e', 's', 'v', 'g', 'r', 'a', 'm', 'a', 'r', 'i', 's', 't'}, output: 10

Design a **recursive** algorithm to solve this problem (15 points).

1. **describe the idea behind your algorithm in English (5 points);**
   1. Set result to false till proven true. Make sure the array has enough space to actually contain the string otherwise the result is false. If the result is possibly in the array check. If there is a value matching return the spot of the first value in the string. Keep going to see if there are any more values present.
2. **provide pseudocode (10 points);** 
   1. Recursive function{if(IndexStart +Stringlength < ArrayLength){

Lastindex = startindex +stringlength-1

ArrayString=Startindex through lastindex

If (ArrayString == String){

Result = Startindex

}

Recursive function(Arrray, Sting, Index Result);

}

Return Result;

}

**Full credit (15 points) will be awarded for a recursive algorithm. Algorithms that are NOT recursive will be scored out of 5 points. For this problem, you do NOT need to analyze the running time.**

1. Write a recursive algorithm that takes as input an array A[] of integers, and gives as output an array B[] where B[i]=A[0]\*A[1]\*...\*A[i].

B[0] = A[0]

B[1] = A[0] \* A[1]

B[2] = A[0] \* A[1] \* A[2]

...

B[i] = A[0] \* A[1] \*...\* A[i]

Design a **recursive** algorithm to solve this problem (15 points).

1. **describe the idea behind your algorithm in English (5 points);**
   1. if there is only one value in the array just return that value. This is the base case. If there are more than one value multiply the last value in the array by the value before, then if there was a value before that multiply that value by the value before, until getting to the first value in the array.
2. **provide pseudocode (10 points);** 
   1. Recursive function{if(i==0){

Return A[i]

}else{

Return (A[i]\*Recursivefunction(A,i-1))

}

}

**Full credit (15 points) will be awarded for a recursive algorithm. Algorithms that are NOT recursive will be scored out of 5 points. For this problem, you do NOT need to analyze the running time.**

**Section 2: Java Implementation**

1. Implement problem 1 in Java (30 points).

Note:

Find a file called Problem1.java in assignment 7 folder.

Complete the method of where().

Test your method in the main method provided following the comments.

**Full credit (30 points) will be awarded for a recursive algorithm. Algorithms that are NOT recursive will be scored out of 10 points.**

1. Implement problem 1 in Java (30 points).

Note:

Find a file called Problem2.java in assignment 7 folder.

Complete the method of recursivearray().

Test your method in the main method provided following the comments.

**Full credit (30 points) will be awarded for a recursive algorithm. Algorithms that are NOT recursive will be scored out of 10 points.**

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt), and to Section 2 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**
2. **Create a folder and name it 'FirstName\_LastName\_assignment\_3'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and submit to iLearn.**