CS2110 Computer Programming Lab
Assignment 6

Date: Sept 14, 2015

Instructions

- In this assignment, you will have to solve 3 problems based on Recursion.
- Read each of the problems carefully before attempting them. You must not make any other assumptions other than those provided for each of the problems.
- Plagiarism of any form from internet sources/peers is strictly prohibited.
- Refer submission guidelines for further information on deadlines for submission.

Problems

Problem 1

In this problem, you are given an array of integers which follow the pattern : they first increase monotonically and then decrease monotonically. Write a C program to output the maximum value for such an array of integers using **Binary Search** algorithm.

You must take input from a file and write output to a file. The input file and output file will be given as arguments to the executable.

Input Format

First line in the input file consists of number of test cases(N). N test cases follow.
Each test case consists of two lines :
First line of each test case is the length of the array.
Second line of test case consists of the array.

Output Format

For each test case in the input file, output the maximum value in a single line in the output file as shown below.

Sample Input	**Sample Output**
2 5 1 8 6 5 4 4 2 1 1 1	8 2

Problem 2

In this problem, you are asked to arrange a set of strings in sorted order. Also, you are provided with a string **OrderString**. This OrderString will specify the priority of the alphabets.

For eg : If the OrderString is "dcfbae", it means $d > c > f > b > a > e$, unlike the lexicographic(dictionary) ordering where we have $a > b > c > d > e > f$.

You can also assume that the strings given *will only contain lower case alphabets* from the OrderString.

Now, based on the priority of the alphabets given in the OrderString, you are required to sort the strings. Write a C program to arrange the given set of strings using **Merge Sort** algorithm.

You must take input from a file and write output to a file. The input file and output file will be given as arguments to the executable.

Input Format

First line in the input file consists of the OrderString.
Next line consists of the number of strings to be arranged(N).
Each of the N lines that follow is a string.

Output Format

Output the strings in the sorted order with one string per each line.

Sample Input	**Sample Output**
dcfbae	
5	deaf
face	cafe
ace	fade
cafe	face
deaf	ace
fade	

Problem 3

In the string world, friendship is valued highly and therefore they are **friends** only if they follow certain conditions. One such criteria is listed below.
Two strings s1 and s2 are **friends iff**

- length(s1) must be equal to length(s2) **and**
- If both are of odd length, then s1 **must** be EQUAL to s2.
- If both are of even length, let s1 = $s_{11}s_{12}$, where s_{11} = first half of s1 and s_{12} = second half of s1. Similarly let s2 = $s_{21}s_{22}$. Then, either (s_{11} , s_{21} are **friends** **<u>and</u>** s_{12}, s_{22} are **friends**) **<u>or</u>** (s_{11}, s_{22} are **friends** **<u>and</u>** s_{12}, s_{21} are **friends**).

Write a C program which uses **recursion** to output whether two strings are **friends** or not. Note that we will test the program with very large strings also, and the brute force approach will not work for such cases. Hence, try to come up with a better algorithm.

You must take input from a file and write output to a file. The input file and output file will be given as arguments to the executable.

Input Format

First line in the input file consists of the number of test cases.
Each of the N lines that follow consist of two strings which must be checked if they are friends or not.

Output Format

Output **1** if strings are friends, else Output **0** in a single line for each of the test cases.

Sample Input	**Sample Output**
2 abcd dcab abcde abdce	1 0

Explanation

For the first test case, abcd dcab - as both are of even length, they are friends iff (ab, dc are friends and cd, ab are friends) or (ab, ab are friends and cd, dc are friends). Observe that ab, ab are friends and dc, cd are friends. Second test case violates the second condition listed above.

Submission Guidelines

Please follow the submission guidelines, not doing so will incur a penalty
Problem 1 must be submitted on or before **20^{th} September 11:30 AM** (i.e in the morning).
You must create a compressed tar file titled `cs14b0xx_lab6_1.tar.gz` replacing `cs14b0xx` in the name with your rollnumber. The tar file on extracting must contain a single folder titled `cs14b0xx_lab6_p1`. This folder must contain all your .c files for problem 1 (you can choose to split it across files or write it in a single file). It must also contain a `Makefile`. Upon running `make`, it must create an executable titled `cs14b0xx_lab6_p1`. We will use this executable and provide input file and output file as arguments to test your output.

Problems 2 and 3 must be submitted on or before **27^{th} September 11:55 PM**.
You must create a compressed tar file titled `cs14b0xx_lab6_2.tar.gz` which on extracting must contain **two folders** titled `cs14b0xx_lab6_p2` and `cs14b0xx_lab6_p3`. Each of these folders must contain all your .c files for the concerned problem (you can choose to split your code for each problem across files or write it in a single file). Each folder must also contain a `Makefile`. Upon running `make` in `cs14b0xx_lab6_p2`, it must create an executable titled `cs14b0xx_lab6_p2` and Upon running `make` in `cs14b0xx_lab6_p3`, it must create an executable titled `cs14b0xx_lab6_p3`. We will use these executables and provide input file and output file as arguments to test your output.

The submitted code should be properly indented and commented. Please make sure you have removed all the temporary files and executables before you make your submission on moodle.