INDIAN STATISTICAL INSTITUTE

Mid-Semestral Examination : 2014–15

Course : Master of Technology in Computer Science (First Year)

Subject : Introduction to Programming (A1) for M.Tech. (CS) I

Date : 15 September 2014 | Maximum Marks : 60 | Duration : 3 Hours

Note: You are expected to code in C to solve the following problem(s), and you are allowed to use any academic resource of your choice during the course of this examination. Write the theoretical part(s) of the solution(s) in the answer-sheet provided, and submit the relevant C source file(s) separately. In the theoretical part(s) of the solution(s), you must provide basic sketch of the algorithm(s) that you use, along with basic complexity analysis of your chosen algorithm(s), wherever appropriate. You must justify the C data-type(s) that you use while coding, and provide basic pictorial illustration(s) of the memory stack-frame(s) for clarification, wherever appropriate.

Problem 1

Maximum Marks: 40 = 10 + 10 + 15 + 5

Write by yourself all required string functions, without using the standard `string.h` library of C.

1A. Accept an English word from the user as a *command-line argument*. If the word is k-letters long, print the following – the original k-letter word, last $(k-1)$ letters of the word, last $(k-2)$ letters of the word, and so on, till the last two letters of the word, and the last letter of the word.

Example Input: `sourav`

Example Output: `sourav`, `ourav`, `urav`, `rav`, `av`, `v`

1B. In continuation to the previous problem, write a function `wordRev` to reverse the original word and store the output as a new word. Write another function `wordRevInp` to reverse this new word, in place, and overwrite itself. Print the following – the original word, the new word formed by applying `wordRev` to the original word, and the overwritten new word after applying `wordRevInp`.

Example Output: `sourav`, `varuos`, `sourav`

1C. In continuation to the previous problem, accept a positive integer r from the user as a *run-time input*. Write a function `wordRot` to right-rotate the original input word by r letters, and store the output as a new word. Write another function `wordRotInp` to right-rotate this new word by r letters, in place, and overwrite itself. Print the following – the original word, the new word formed by applying `wordRot` to the original word, and the overwritten new word after applying `wordRotInp`.

Example Input: `2`

Example Output: `sourav`, `avsour`, `uravso`

1D. In continuation to the previous problem, print all possible right-rotations of the original word.

Example Output: `sourav`, `vsoura`, `avsour`, `ravsou`, `uravso`, `ouravs`

Problem 2

Maximum Marks: 20 = 10 + 10

An *anagram* is a type of word play, where the result of rearranging the letters of a word, using all the original letters exactly once, produce a new word that belongs to some predefined dictionary.

2A. Accept two lowercase English words from the user as *command-line arguments*. Check if one word is an *anagram* of the other, and print `Anagrams` or `NOT Anagrams` accordingly.

Example Input: `stop post`

Example Output: `Anagrams`

Example Input: `stop sign`

Example Output: `NOT Anagrams`

2B. Accept an arbitrary list of lowercase English words from the user as *command-line arguments*, and store them as the predefined *dictionary*. Accept another lowercase English word from the user as a *run-time input*, and determine if any *anagram* of this word exists in the aforesaid dictionary. If the answer is affirmative, print all anagrams of the word from the dictionary.

Example Input: {`stop block sign post club fly run dance`} and `tops`

Example Output: `Anagrams found : stop post`

Example Input: {`stop block sign post club fly run dance`} and `sourav`

Example Output: `Anagrams found : None!`

Solving a problem correctly is necessary, but not sufficient, as it does not guarantee the maximum marks alloted for the problem. Sufficient credit is reserved in each case for *smart* algorithm and *good* coding practice. Good luck! ☺