BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
HYDERABAD
CAMPUS,
Data Structures and Algorithms
CS F211 / IS F211

Homework Assignment – 5

1. Write an algorithm called Longest-Repeated-Substring (T) that takes a string T representing some text, and finds the longest string that occurs at least twice in T. If no such substring exist, then the algorithm returns "No substring found"
   Input:
   a b c a c a n a g r t m m t t k k c a c a n a g r a g c a d d c c
   Output:
   c a c a n a g r

2. Design an efficient in-place algorithm called Partition-Even-Odd (Arr) that partitions an array **Arr** in even and odd numbers. The algorithm must terminate with **Arr** containing all its even elements preceding all its odd elements.
   For example: Arr = 7, 17, 74, 21, 7, 9, 26, 10
   The result might be Arr = 74, 26, 10, 21, 7, 9, 17, 7

3. Write an algorithm called OCCURRENCES that, given an array of numbers **A**, prints all the distinct values in **A** each followed by its number of occurrences. The algorithm may modify the content of **A**, but may not use any other memory. Each distinct value must be printed exactly once. Values may be printed in any order.
   For example, if A = 28, 1, 0, 1, 0, 3, 4, 0, 0, 3
   Output: 28 1; 1 2; 0 4; 3 2; 4 1

4. Write an in-place partition algorithm called Modulo-Partition (**A**) that takes an array **A** of **n** numbers and changes **A** in such a way that (1) the final content of **A** is a permutation of the initial content of **A**, and (2) all the values that are equivalent to **0 mod 10** precede all the values equivalent to **1 mod 10**, which precede all the values equivalent to **2 mod 10**, etc.
   For example, A = 7, 62, 5, 57, 12, 39, 5, 8, 16, 48
   A = 12, 62, 5, 5, 16, 57, 7, 8, 48, 39

5. Implement a dictionary data structure that supports longest prefix matching. Specifically, write the following two algorithms:
   - Build-Dictionary(W) takes a list W of n strings and builds the dictionary
   - Longest-Prefix(k) takes a string k and returns the longest prefix of k found in the dictionary, or NULL if none exists. The time complexity of Longest-Prefix(k) must be Little oh (o(n)), that is, sublinear in the size n of the dictionary.

For example, assuming the dictionary was built with strings, “luna”, “lunatic”, “a”, “al”, “algo”, “an”, “anto”, then if k is “algorithms”, then Longest-Prefix(k) should return “algo”, or if k is “anarchy” then Longest-Prefix(k) should return “an”, or if k is “lugano” then Longest-Prefix(k) should return NULL.

6. Students attending course of Machine Learning are categorized in three groups Gp1, Gp2 and Gp3 by some classifier algorithms. Student have attributes like name, branch, cgpa, year, group. You are given list of students and you have to arrange the students in a fashion such that student who belong to Gp1 comes first then comes Gp2 and then Gp3(In case any 2 students s1, s2 belong to same group then the student who is coming first in original list must come before in your output list i.e. stable property). You have to solve this problem in O(n) and O(1) (= constant) space complexity. Define structure student with relevant attributes. And output the list with name along with group to which he belongs.

   Input:

   N

   Following N details of students

7. Game x is played in a country y. There are N states in country y and you are given the number of players from each of them. There is a rule that for any team to participate in that game, there should be exactly M players in a team and no two player should be from same state. How many maximum teams can you form?

   Constraints: 1 <= N <= 50, 2 <= M <= 100, 1 <= nop[i] <= 109

   Input:

   N M

   Second line contains the number of players from ith state

   Sample Input-1:

   4 3

   2 4 3 6

   Output:

   4

   Explanation:

   N = 4 and M = 3

   Let 4 states be s1, s2, s3, s4. Given that we have 2 from s1, 4 from s2, 3 from s3 and 6 from s4. Each team should contain 3 players from different states. We can form the following team: {s1, s2, s3}, {s1, s2, s4}, {s2, s3, s4}, {s2, s3, s4}. Note that even though there are 15 players in total we can't form more than 4 teams.

Sample Input-2:

6 4

10 11 12 13 14 15

Output:

18

8. Krish has come up with a formula of sorting the list of names(max character is 50) because his understanding of world is different. Here is his formula of sorting the names: Take the name and find the sum of ascii value for each character and then convert the final sum it to binary representation and count how many 1s are there in the final sum and then arrange the name in ascending order of the total number of 1s in the final sum. In case 2 names have same total number of 1s then name coming first in the list should come first in the output list i.e. stable property. You all being computer geeks, help Krish to solve this problem in O(n) time. So read list of string and sort it in Krish-way.

9. Given an array of 0s and 1s, find the position of 0 to be replaced with 1 to get longest continuous sequence of 1s. Expected time complexity is O(n) and auxiliary space is O(1).

   Example:

   Input: arr[] = {1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1}

   Output: Index 9

   Assuming array index starts from 0, replacing 0 with 1 at index 9 causes the maximum continuous sequence of 1s.

   Input: arr[] = {1, 1, 1, 1, 0}

   Output: Index 4

10. You are given a dictionary file containing words, each in separate lines. Assume each word is less than 50 letters and each letter is among a, b, c,…, z. So, “apple” upon sorting becomes “aelpp”. Many words in the dictionary will result in the same string after sorting. For example, sorted string of both “cat” and “act” will be “act”. Similarly, sorted string of both “listen” and “silent” is “eilnst”. These actual words are anagrams whose sorted string is same. You have to write a program to read each word from a file wordfile.txt (create it and write some words), sort the word in dictionary order, and print the sorted word and actual word. Use Unix pipes for redirecting outputs. The above print would actually be piped to sort utility. The output of sort in turn would be piped to the second program you need to write (look at the command at the end).

So, you need to write two programs:

a. "*sortwords.c*" that reads file and sorts each word, and then prints sorted string and the actual word in a line.

printf("%s %s\n", sortedword, actualword);

Call this file sortwords.c. Compile and create executable from this file called sortwords. The above program's output can be fed to *sort* program of Unix. (see the command at the end).

b. The second program *anagrams.c* (its executable called print_anagrams) that will get the input from the *sort* program (*sort* program sorts all the lines of input line by line, so each anagram will become contiguous after *sort*) The output from the *sort* program is taken by program (called) anagrams, and it should print all the anagrams in a separate line.

Save the final output file as *anagrams.txt*.

The **command** that you will run finally will look like:
$ ./sortwords < wordfile.txt | sort | ./print_anagrams > anagrams.txt

An example follows:
wordfile.txt contains
boy
stray
act
tops
apple
cat
stop

The following is the output of sortwords:
act act
apple aelpp
boy boy
act cat
opst stop
arsty stray
opst tops

On sorting (using sort utility in Unix):
act act
act cat
aelpp apple
arsty stray

boy boy
opst stop
opst tops

The above would then be used by ./print_anagrams to print all anagrams in a single line. The output of ./anagrams would be sent to file anagrams.txt which should now contain:
act cat
apple
stray
boy
stop tops

Note : See that each line contains actual words that are anagrams.

----------XXXXXXXXXXXXXXXXXXXXXXXXX----------