BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
HYDERABAD
CAMPUS,
Data Structures and Algorithms
CS F211 / IS F211
Homework Assignment – 9

1. Your friend is writing characters on a paper, but he keeps forgetting the previous characters he has written. So he doesn't know if he has written a particular character in past or not. This results in many duplicate characters. You are allowed only two pass through this text and you have to remove the duplicate characters and get the expected text which your friend have intended to write.
   Ex :
   text is : agha#jijk!lgk!kkagg
   output is : agh#jik!l

2. Suppose you are given a text without spaces. You are asked to find the length and the substring representing the longest substring without repeating characters in it in O(n) time. If there are multiple substring print any one (even printing all substrings is fine).
   For ex:
   text : abcddaabdccfg
   output : max length is 4 and strings are abcd and abdc

3. In Japan lot of people in morning while drinking tea solve the Sudoku puzzle. One morning when Japanese Prime Minister came to India, he was solving Sudoku and after solving it he gave it to our Prime Minister and asked to verify whether he has solved it correctly or not. And now Prime Minister of India needs your help. This is a matter of pride for you and you pledge to help him to find out if Sudoku solved is correctly or not.
   Input : 9 X 9 2D array representing the solution of Sudoku
   Output : YES or NO

4. You are given a text **T** with no spaces and separate list of **n** words w_1, w_2, … w_n that are of **same** length. Words can repeat. You have to find and print all the smallest substrings along with starting and ending indexes in T which contains all the n words in any order and there should not be any characters between those words in the substring i.e no intervening character between words.
   For ex :
   Text : indiabaratgreatwelldonebaratgreatindiacountry
   Words : 3 words : great, india, barat

   Output : substring are :
   indiabaratgreat , startIndex : 0 endIndex : 14
   baratgreatindia , startIndex : 23 endIndex:37

5. Karna was fighting a great war and has to break **N** big mayavi walls coming one by one to him as he was moving ahead to win the war. These mayavi walls can only be broken by arrows of specific type $\mathbf{W_i}$. But as soon as Karna reaches the Wall, a Demi-God appears and presents him an arrow of some type $\mathbf{A_i}$. If this arrow $\mathbf{A_i}$ is same as the arrow type $\mathbf{W_j}$ needed to break the Wall he will use it and break the wall else he will keep it with himself.
   Help Karna find the minimum number of arrows he must carry with himself to be able to win the war.

Input:
N is the number of walls followed by N lines containing A_i W_i.

Ai : Number representing Arrow Type given by Demi-God just before every wall.
Wi : Number representing Arrow Type which can break the i^{th} Wall.

Constraints :
$1 \leq N \leq 1000$
$0 \leq Ai , Wi \leq 10000$

For ex:
Input:
6
10 5
5 10
4 4
1 3
2 3
3 1

Output :
3

6. **Write a menu driven C program to implement a** hash table of size 10. Consider input keys {71, 23, 73, 99, 44, 19, 49, 93, 81, 39} using the hash function h (k) = k mod 10. Upon collisions use chaining ***(linked-list)***. Your program must support the following operations.
    a. Insert an element
    b. Print the hash table
    c. Searching an element (i.e., search (key))
    d. Delete an element
    e. Exit
7. **Write a menu driven C program to implement a** hash table of size 10. Consider input keys {71, 23, 73, 99, 44, 19, 49, 93, 81, 39} using the hash function h (k) = k mod 10. Upon collision you use another hashing function h'(k) = 7 − k mod 7, and on further collisions use chaining ***(use a doubly-linked list).*** Your program must support the following operations.
    a. Insert an element
    b. Print the hash table
    c. Searching an element (i.e., search (key))
    d. Delete an element
    e. Exit
8. **Write a menu driven C program to implement a** hash table of size 10. Consider input keys {71, 23, 73, 99, 44, 19, 49, 93, 81, 39} using the hash function h (k) = k mod 10. Upon collisions use linear probing (h (k) = (k+i) mod 10). Your program must support the following operations.
    a. Insert an element
    b. Print the hash table
    c. Searching an element (i.e., search (key))
    d. Delete an element
    e. Exit

9. **Write a menu driven C program to implement a** hash table of size 10. Consider input keys {71, 23, 73, 99, 44, 19, 49, 93, 81, 39} using the hash function h (k) = k mod 10. Upon collisions use quadratic probing (h (k) = $(k+i^2)$ mod 10). Your program must support the following operations.
    a. Insert an element
    b. Print the hash table
    c. Searching an element (i.e., search (key))
    d. Delete an element
    e. Exit

10. You have a dictionary of English words and are asked to find anagrams of a given word. Anagrams are dictionary words that contain same letters but in different permutations.
    e.g.one, eon, neo are anagrams.
    A **word** is **not considered anagram of itself**.
    Your task is to write a program using hash table, which, when run, will ask the user continuously to enter a word as input, and will print output in format given in example below, until the user wants to quit by pressing 'q'. The output is: "given word: number of anagrams followed by the words that are anagrams of the given word".
    So in the above example, if 'one' is given as input by the user the output should exactly be:
    one: 2 eon neo

    The **program** should take as its **argument** the **word file and the size of hash table**. The **size** of hash table given as argument to the program can be any **positive integer**. Experiment with different values.

    A sample run of the program:

    $./a.out DictionaryFile.txt 100000

    Enter a word(q to exit): **one**
    one: 2 eon neo

    Enter a word(q to exit): bgggg
    bgggg: 0

    Enter a word(q to exit): **neo**
    neo: 2 eon one

    Enter a word(q to exit): q
    $
    Note: Use the dictionary file DictionaryFile.txt provided on CMS.

----------XXXXXXXXXXXXXXXXXXXXXXXXXX----------