Data Structures and Algorithms (CS F211)
Second Semester 2018-2019
Lab Sheet 3

1. You are given two strings ***t*** and ***p***. The string ***t*** contains all the letters of string ***p*** in the same order (it doesn't have to be contiguous). For example, ***t*** could be 'abdefcb' and ***p*** could be 'abb' but ***p*** could not be 'bab' (order is not preserved) or 'bxy' ('xy' is not in ***t***). You are also given a set of operations on an array ***a***. Each element ***a***[i] denotes the index of the letter to be removed from string ***t*** (**a**[i]>=1). For example, if the array is '3 5 4 7 2 1 6' applied on the string ***t*** which is 'abdefcb', then first remove the 3rd element 'd', so ***t*** becomes 'ab~~d~~efcb' (character 'd' has been striked through). Then we remove the 5th element 'f', so ***t*** becomes 'ab~~d~~e~~f~~cb'(character 'f' has been striked through) and so on. Note that after removing one letter, the indices of other letters don't change. So, if a letter was at 5th position and you remove the 3rd letter, then still the letter at the 5th position stays at 5th position.
Your task is to determine the maximum number of letters that can be removed from string ***t*** following the operations in the array so that string ***p*** is obtainable.
Also note that array ***a*** has the same size as the string ***t***. So, by the end, after all operations in array ***a*** have been executed you should not be left with any unprocessed element in string ***t***, as the elements in array ***a*** are all unique. Constraints: $|t| \leq 1000$, $|p| \leq 1000$.

Input:
First line contains string ***t*** and second line contains string ***p***.
Third line contains an integer ***n*** which denotes the number of elements in array ***a***. The size of ***a*** is equal to the size of string ***t***.
Fourth line contains array ***a***.

Output:
Print a single integer number, the maximum number of letters that you can remove following the order in array ***a***.

Example :
abdefcb
abb
7
3 5 4 7 2 1 6

Output :
3

Explanation:
Following the order of removals in array ***a***, you first remove 'd', then 'f', then 'e'. Now if you remove the 7th letter which is 'b', then abb is in no way obtainable. So, you stop after 3 operations and output is 3.

2. You are given a string ***s*** and ***m*** queries. Each of the ***m*** queries contains 3 integers f, r, k. Each of the query corresponds to cyclically shifting the values from ***s***[f] to ***s***[r] for k times. One operation of a cyclic shift (rotation) is equivalent to moving the last character to the position of the first character and shifting all other characters one position to the right. Assume index starts from 1. For example, if the string ***s*** is 'abbcde' and the query is 2, 4, 1, then the values from ***s***[2] to ***s***[4] are 'bbc' and shifting it k = 1 times results in 'cbb'. The final string is 'acbbde'.

Input:
The first line contains string ***s*** and the second line contains a single integer ***m*** denoting the number of queries. Each of the next ***m*** lines contains f, r, k denoting the bounds of each operation. ($1 \leq f \leq r \leq |s| \leq 1000$, $1 \leq k \leq 1000$).

Ouput:
A single string which is the result of all the operations.

Example:
abbcde
2
2, 4, 1
1, 3, 2
Output:
cbabde

3. Given a singly linked list, write a program to shuffle it according to the following rule: segregate all the elements in such a way that elements at positions k*i (where i starts from 1 and k<=1000, k may not be less than the number of elements present in the linked list, k may not be a multiple of the number of elements present in the linked list) appear before the rest of the elements and the remaining elements are appended in reverse order.
Input:
The first line contains the value of k.
The second line contains the elements of the linked list (The total number of elements present in the linked list can't be taken as an input).
Output:
An updated linked list based on the segregation rule described in the problem.
Example:
3
1 2 3 4 5 6 7 8 9
Output:
3 6 9 8 7 5 4 2 1

4. You are living in a country that has n cities (numbered from 1 to n) connected to each other using n-1 bidirectional roads. No city is completely isolated from all the other cities. Each road connects two cities. (Represent this scenario as an undirected, acyclic, connected graph). You love walking and you plan to walk between two cities (u, v), taking the shortest path from u to v.
There are 2 special cities that you dislike visiting namely (x, y). You will avoid all possible paths, in which you first encounter x and then y. The other permutation (first y then x) is allowed. Individual cities x and y appearing separately in the route are fine too. Find out the total number of possible (u, v) pairs such that you do not encounter first the city x then the city y in your route while you are travelling from city u to city v.

Input:
First line contains n, x, y.
Each of the next n-1 lines contains two integers, denoting that these two cities are connected.
Output:
A single integer denoting the total number pairs of cities that follow the restriction.
Example:
3 1 3
1 2
2 3
Answer:
5
Explanation: Path 1 (1 to 2), Path 2 (2 to 1), Path 3 (2 to 3), Path 4 (3 to 2) and Path 5 (3 to 2 to 1). The Path (1 to 2 to 3) is not considered because of the restriction.

5. You are given an N*N matrix filled with 0s and 1s. You have to find out the largest region of 1s in the given matrix. A region is a set of connected 1s. Two 1s are connected if they are adjacent to each other horizontally,

vertically, diagonally. **Constraint:** N <= 1000. You have to only print the size of the largest region, i.e., the total number of 1s present in it.

Input:
First line contains N.
Second line contains elements of the N*N matrix.
Output: Largest region of 1s
Example:
5
0 0 0 0 0
0 1 1 1 0
0 1 1 0 0
1 0 0 0 1
0 0 0 0 0
Output: 6.

Explanation:
0 0 0 0 0
0 1 1 1 0
0 1 1 0 0
1 0 0 0 1
0 0 0 0 0

The red 1s form the largest region. The blue 1 is not a part of the largest region since wrapping around is not considered in any of the directions.

6. Steph Curry is stuck at the horror house with a string of English lower-case characters as the only weapon. After searching the house, he finds a gate that can only be opened by a string that is an anagram of a palindrome of English lower-case characters. Check if Steph can come out of the horror house or not.

Constraint:
Length of the string <= 10
Input:
Given a string s

Sample Input 1:
aabaa
Sample Output 1:
Yes
Sample Input 2:
abaaa
Sample Output 2:
Yes
Sample Input 3:
a
Sample Output 3:
Yes
Sample Input 4:
aaba

Sample Output 4:
No

7. Steph fortunately gets out of the horror house using his string. The gate keeper did not like this. He is a big fan of Caesar. He asks you to return the Caesar Cipher of the given password. Caesar Cipher increases the ASCII value of the given letter by 3 and returns the letter corresponding to the new ASCII value. If the ASCII value exceeds the value of z, then start again from the beginning of the lower-case letters.

Constraint:
Length of the string < 100
Input Format:
Given a string s
Sample Input 1:
abcdefghijklmnopqrstuvwxyz
Sample Output 1:
defghijklmnopqrstuvwxyzabc
Sample Input 2:
abaaa
Sample Output 2:
deddd
Sample Input 3:
a
Sample Output 3:
d
Sample Input 4:
aaba
Sample Output 4:
dded

8. The advanced Atlantean city has N buildings. Our superhero Aquaman needs to go from one building in the city to another. The only way Aquaman can move in the city is through a network of M sharks. Every Shark can only move from one building to another. Given a query of the form (x, y), find if Aquaman can reach building y from building x using the network of M sharks. You may be given t such queries. Please note that if a Shark can move from building x to building y that doesn't imply that it can also move from building y to building x.

Constraint:
1 <= N <= 1000
1 <= M <=1000
1 <= t <= 10
1<=x<=N
1<=y<=N
Input:
The first line contains two integers in the order N, M
Given M lines with two building numbers a, b. Each line i represents that the i^{th} shark can move from building a to building b
This line contains value of the number of queries t
Given t lines with two building numbers x, y

Sample Input:

5 6
1 4
1 2
1 3
4 2
4 3
5 1
3
3 5
4 3
5 3

Sample Output:
No
Yes
Yes

9. Given a directed graph with N vertices and M edges with no self-loops. Find all the vertices from where all the other vertices can be reached. If no such vertex exists, print the message “no” or else print “yes” and the vertex numbers.

Constraint:
1 <= N <= 1000
1 <= M <=1000

Input:
The first line contains two integers in the order N, M
Given M lines with two vertex numbers x, y, representing an edge from vertex x to vertex y

Sample Input:
5 7
1 2
1 4
1 3
4 2
4 3
3 5
5 1

Sample Output:
Yes, 1,3,4,5

****