L10 : Character Arrays and 2d Arrays Practice Questions

1-Tut : Check Palindrome

Send Feedback

Given a string, determine if it is a palindrome, considering only alphanumeric characters.

Palindrome

A palindrome is a word, number, phrase, or other sequences of characters which read the same backwards and forwards.

Example:

If the input string happens to be, "malayalam" then as we see that this word can be read the same as forward and backwards, it is said to be a valid palindrome.
The expected output for this example will print, 'true'.
From that being said, you are required to return a boolean value from the function that has been asked to implement.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

Output Format:

The only line of output prints either 'true' or 'false'.

Note:

You are not required to print anything. It has already been taken care of.

Constraints:

0 <= N <= 10^6
Where N is the length of the input string.
Time Limit: 1 second

Sample Input 1 :

abcdcba

Sample Output 1 :

true

Sample Input 2:

coding

Sample Output 2:

false

```
1.  bool checkPalindrome(char str[]) {
2.      // Write your code here
3.      int len = 0;
4.      while(str[len] != '\0'){
5.          len++;
6.      }
7.      int i = 0, j = len-1;
8.      while(i < j){
```

```
9.
10.         if(str[i] == str[j]){
11.             i++;
12.             j--;
13.             continue;
14.         }
15.         else if(str[i] != str[j])
16.         {
17.             return 0;
18.         }
19.     }
20.     return 1;
21. }
```

2-Tut : Replace Character

Send Feedback

Given an input string S and two characters c1 and c2, you need to replace every occurrence of character c1 with character c2 in the given string.

Input Format :

Line 1 : Input String S
Line 2 : Character c1 and c2 (separated by space)

Output Format :

Updated string

Constraints :

1 <= Length of String S <= 10^6

Sample Input :

abacd
a x

Sample Output :

xbxcd

```
1.  void replaceCharacter(char input[], char c1, char c2) {
2.      // Write your code here
3.      int i = 0;
4.      while(input[i] != '\0'){
5.
6.          if(input[i] == c1){
7.              input[i] = c2;
8.          }
9.
10.         i++;
11.     }
12. }
```

3-Tut : Trim Spaces

Send Feedback

Given an input string S that contains multiple words, you need to remove all the spaces present in the input string.

There can be multiple spaces present after any word.

Input Format :

String S

Output Format :

Updated string

Constraints :

1 <= Length of string S <= 10^6

Sample Input :

abc def g hi

Sample Output :

abcdefghi

```
1.  void trimSpaces(char input[]) {
2.      // Write your code here
3.      int i = 0 , j = 0;
4.      while(input[i] != '\0'){
5.
6.          if(input[i] != ' '){
7.              input[j++] = input[i];
8.
9.          }
10.
11.         i++;
12.     }
13.
14.     input[j] = '\0';
15. }
```

4-Tut : Reverse Word Wise

Send Feedback

Reverse the given string word wise. That is, the last word in given string should come at 1st place, last second word at 2nd place and so on. Individual words should remain as it is.

Input format :

String in a single line

Output format :

Word wise reversed string in a single line

Constraints :

0 <= |S| <= 10^7

where |S| represents the length of string, S.

Sample Input 1:

Welcome to Coding Ninjas

Sample Output 1:

Ninjas Coding to Welcome

Sample Input 2:

Always indent your code

Sample Output 2:

code your indent Always

```
void reverseStringWordWise(char input[]) {
    // Write your code here
    int len = 0;
    int i = 0;
    while(input[i] != '\0'){
        len++;
        i++;
    }

    i = 0;
    int j = len - 1;

    // Reverse the string
    while(i < j){
        char temp = input[i];
        input[i] = input[j];
        input[j] = temp;
        i++;
        j--;
    }

    i = 0;
    j = 0;
    while(input[i] != '\0')
    {

                int k = 0;
                if(input[i] == ' ')
                {
                    k = i-1;
                    while(j < k){
                      char temp = input[j];
                       input[j] = input[k];
                       input[k] = temp;
                       j++;
                       k--;
                      }
```

```
                    j = i+1;

                            }

        i++;
    }

    i = len -1;

    while(j < i){

                     char temp = input[j];
                      input[j] = input[i];
                      input[i] = temp;
                      j++;
                      i--;

    }

}
```

5-Tut : Print All Substrings

Send Feedback

For a given input string(str), write a function to print all the possible substrings.

Substring

A substring is a contiguous sequence of characters within a string.
Example: "cod" is a substring of "coding". Whereas, "cdng" is not as the characters taken are not contiguous

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

Output Format:

Print the total number of substrings possible, where every substring is printed on a single line and hence the total number of output lines will be equal to the total number of substrings.

Note:

The order in which the substrings are printed, does not matter.

Constraints:

0 <= N <= 10^6
Where N is the length of the input string.
Time Limit: 1 second

Sample Input 1:

abc

Sample Output 1:

a
ab
abc

b
bc
c

Sample Input 2:

co

Sample Output 2:

c
co

o

```
1.  void printSubstrings(char input[]) {
2.      // Write your code here
3.
4.      for(int i = 0; input[i] != '\0'; i++)
5.
6.      {
7.
8.          for(int j = i; input[j] != '\0'; j++){
9.
10.             for(int k = i; k <= j; k++){
11.                 cout << input[k];
12.             }
13.             cout<< endl;
14.         }
15.           cout<< endl;
16.
17.     }
18. }
```

6-Tut : Column Wise Sum

Send Feedback

Given a 2D integer array of size M*N, find and print the sum of ith column elements separated by space.

Input Format :

First and only line of input contains M and N, followed by M * N space separated integers representing the elements in the 2D array.

Output Format :

Sum of every ith column elements (separated by space)

Constraints :

1 <= M, N <= 10^3

Sample Input :

4 2 1 2 3 4 5 6 7 8

Sample Output :

16 20

```
#include<iostream>
using namespace std;
int main(){

        /*  Read input as specified in the question.
             * Print output as specified in the question.
             */
   int m ,n;
     cin >> m >> n;
    int a[m][n];

    for(int i = 0; i<m ; i++){

       for(int j = 0; j < n; j++){
           cin >> a[i][j];
       }

    }

    for(int j = 0; j < n; j++){
      int sum = 0;
       for(int i = 0; i< m; i++){
           sum = sum + a[i][j];
       }
       cout << sum << " ";
    }
}
```

7-Tut : Largest Row or Column

Send Feedback

For a given two-dimensional integer array/list of size (N x M), you need to find out which row or column has the largest sum(sum of all the elements in a row/column) amongst all the rows and columns.

Note :

If there are more than one rows/columns with maximum sum, consider the row/column that comes first. And if ith row and jth column has the same largest sum, consider the ith row as answer.

Input Format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.
First line of each test case or query contains two integer values, 'N' and 'M', separated by a single space. They represent the 'rows' and 'columns' respectively, for the two-dimensional array/list.
Second line onwards, the next 'N' lines or rows represent the ith row values.
Each of the ith row constitutes 'M' column values separated by a single space.

Output Format :

For each test case, If row sum is maximum, then print: "row" <row_index> <row_sum>
OR
If column sum is maximum, then print: "column" <col_index> <col_sum>

It will be printed in a single line separated by a single space between each piece of information.
Output for every test case will be printed in a seperate line.

Consider :

If there doesn't exist a sum at all then print "row 0 -2147483648", where -2147483648 or -2^31 is the smallest value for the range of Integer.

Constraints :

1 <= t <= 10^2
0 <= N <= 10^3
0 <= M <= 10^3
Time Limit: 1sec

Sample Input 1 :

1
2 2
1 1
1 1

Sample Output 1 :

row 0 2

Sample Input 2 :

2
3 3
3 6 9
1 4 7
2 8 9
4 2
1 2
90 100
3 40
-10 200

Sample Output 2 :

column 2 25

column 1 342

```
#include <bits/stdc++.h>
void findLargest(int **input, int nRows, int mCols)
{
    //Write your code here
    if(nRows == 0 && mCols == 0){
        cout << "row" << " " << 0 <<" " << INT_MIN;
        return;
    }

    int row[nRows] = {0};
    int col[mCols] = {0};

    for(int i = 0; i < nRows; i++){
        int rsum = 0;

```

```
16.         for(int j = 0; j < mCols; j++){
17.             rsum = rsum + input[i][j];
18.         }
19.         row[i] = rsum;
20.
21.     }
22.
23.     for(int j = 0; j < mCols; j++){
24.         int colsum = 0;
25.
26.         for(int i = 0; i < nRows; i++){
27.             colsum = colsum + input[i][j];
28.         }
29.         col[j] = colsum;
30.
31.     }
32.     int rmax = row[0], rindex = 0,rcount = 0;
33.     int colmax = col[0], colindex = 0,colcount = 0;
34.
35.     for(int i = 0; i < nRows; i++){
36.
37.         if(row[i] > rmax ){
38.             rmax = row[i];
39.             rindex = i;
40.         }
41.     }
42.
43.      for(int i = 0; i < mCols; i++){
44.
45.         if(col[i] > colmax ){
46.             colmax = col[i];
47.             colindex = i;
48.         }
49.     }
50.
51.     if(rmax > colmax){
52.         cout << "row" << " " << rindex << " " << rmax;
53.     }
54.     if(rmax < colmax){
55.         cout << "column" << " " << colindex << " " << colmax;
56.     }
57.
58.     if(rmax == colmax){
59.         cout << "row" << " " << rindex << " " << rmax;
60.     }
61.
62.
63. }
```

8-Tut : Wave Print

Send Feedback

For a given two-dimensional integer array/list of size (N x M), print the array/list in a sine wave order, i.e, print the first column top to bottom, next column bottom to top and so on.

Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.
First line of each test case or query contains two integer values, 'N' and 'M', separated by a single space. They represent the 'rows' and 'columns' respectively, for the two-dimensional array/list.
Second line onwards, the next 'N' lines or rows represent the ith row values.
Each of the ith row constitutes 'M' column values separated by a single space.

Output format :

For each test case, print the elements of the two-dimensional array/list in the sine wave order in a single line, separated by a single space.
Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= N <= 10^3
0 <= M <= 10^3
Time Limit: 1sec

Sample Input 1:

```
1
3 4
1  2  3  4
5  6  7  8
9 10 11 12
```

Sample Output 1:

```
1 5 9 10 6 2 3 7 11 12 8 4
```

Sample Input 2:

```
2
5 3
1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
3 3
10 20 30
40 50 60
70 80 90
```

Sample Output 2:

```
1 4 7 10 13 14 11 8 5 2 3 6 9 12 15

10 40 70 80 50 20 30 60 90
```

```
1.  void wavePrint(int **input, int nRows, int mCols)
2.  {
3.      //Write your code here
4.      for(int j = 0; j < mCols; j++){
5.          //int k = j;
6.
7.                      if(j % 2 == 0){
8.
9.                         for(int i = 0; i < nRows ; i++){
10.                           cout << input[i][j] << " ";
11.                        }
12.                     }
13.                     else {
14.                           for(int i = nRows-1; i >= 0 ; i--){
15.                              cout << input[i][j] << " ";
16.                           }
17.
18.                     }
19.     }
20. }
```

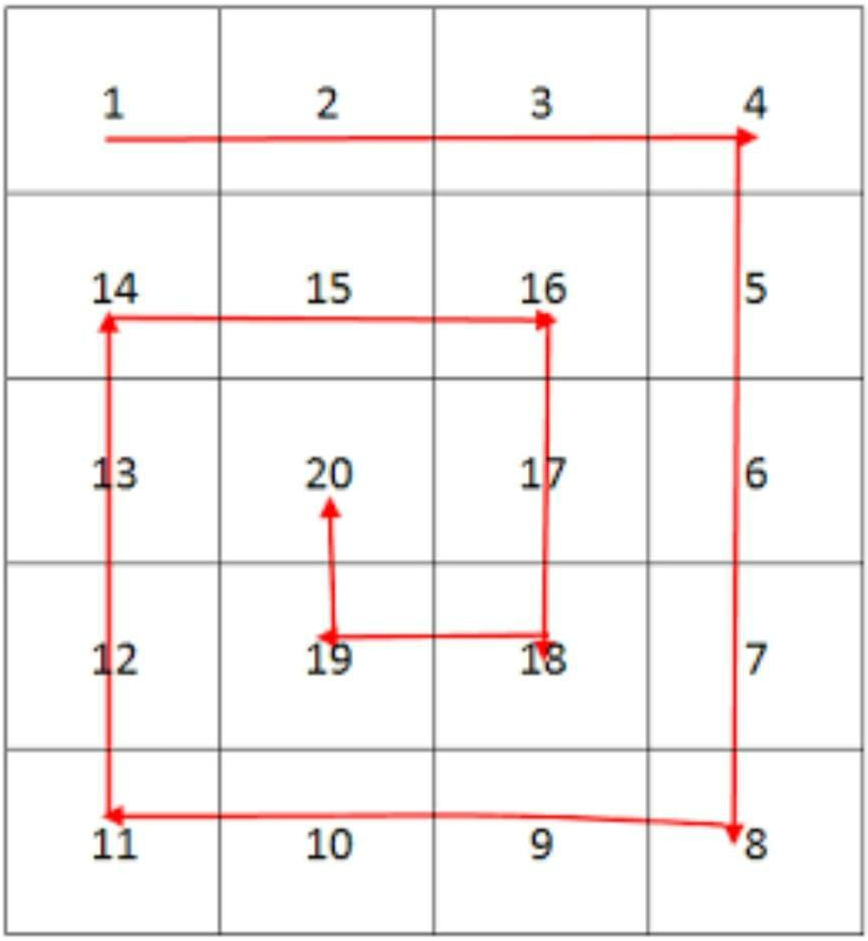
9-Tut : Spiral Print

Send Feedback

For a given two-dimensional integer array/list of size (N x M), print it in a spiral form. That is, you need to print in the order followed for every iteration:

a. First row(left to right)
b. Last column(top to bottom)
c. Last row(right to left)
d. First column(bottom to top)
Mind that every element will be printed only once.

Refer to the Image:



Output :	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

Input format :

The first line contains an Integer 't' which denotes the number of test cases or queries to be run. Then the test cases follow.

First line of each test case or query contains two integer values, 'N' and 'M', separated by a single space. They represent the 'rows' and 'columns' respectively, for the two-dimensional array/list.

Second line onwards, the next 'N' lines or rows represent the ith row values.

Each of the ith row constitutes 'M' column values separated by a single space.

Output format :

For each test case, print the elements of the two-dimensional array/list in the spiral form in a single line, separated by a single space.

Output for every test case will be printed in a seperate line.

Constraints :

1 <= t <= 10^2
0 <= N <= 10^3
0 <= M <= 10^3
Time Limit: 1sec

Sample Input 1:

```
1
4 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Sample Output 1:

```
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
```

Sample Input 2:

```
2
3 3
1 2 3
4 5 6
7 8 9
3 1
10
20
30
```

Sample Output 2:

```
1 2 3 6 9 8 7 4 5

10 20 30
```

```
1.  void spiralPrint(int **input, int nRows, int nCols)
2.  {
3.      //Write your code here
4.
5.      int rs = 0, re = nRows-1;
6.      int cs = 0, ce = nCols -1;
7.      int count = 0,total = nRows * nCols;
8.      while(count < total){
9.
10.         // 1.
11.         for(int i = cs; i <= ce; i++){
12.             if(count == total){
13.                 break;
14.             }
15.
16.             cout << input[rs][i] << " ";
17.             count++;
18.
19.         }
```

```
        rs ++;

        // 2.
        for(int i = rs; i<= re; i++){
            if(count == total){
                break;
            }
            cout << input[i][ce] << " ";
            count++;

        }
        ce--;
        // 3

        for(int i = ce; i >= cs; i--){
           if(count == total){
                break;
            }

            cout << input[re][i] << " ";
            count++;
        }


        re--;

        // 4


        for(int i = re; i >= rs; i--){

            if(count == total){
                break;
            }
            cout << input[i][cs]<<" ";
            count ++;
        }

        cs ++;


    }

}
```

10-Ass : **Check Permutation**

Send Feedback

For a given two strings, 'str1' and 'str2', check whether they are a permutation of each other or not.

Permutations of each other

Two strings are said to be a permutation of each other when either of the string's characters can be rearranged so that it becomes identical to the other one.

Example:

str1= "sinrtg"

str2 = "string"

The character of the first string(str1) can be rearranged to form str2 and hence we can say that the given strings are a permutation of each other.

Input Format:

The first line of input contains a string without any leading and trailing spaces, representing the first string 'str1'.

The second line of input contains a string without any leading and trailing spaces, representing the second string 'str2'.

Note:

All the characters in the input strings would be in lower case.

Output Format:

The only line of output prints either 'true' or 'false', denoting whether the two strings are a permutation of each other or not.

You are not required to print anything. It has already been taken care of. Just implement the function.

Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

Sample Input 1:

abcde

baedc

Sample Output 1:

true

Sample Input 2:

abc

cbd

Sample Output 2:

false

```
1.  bool isPermutation(char input1[], char input2[]) {
2.      // Write your code here
3.
4.      int freq[256] = {0,0};
5.
6.      if(strlen(input1) != strlen(input2) ){
7.          return 0;
8.      }
9.
10.     int i = 0;
11.     while(input1[i] != '\0'){
12.         int c = input1[i];
13.         freq[c] = freq[c] + 1;
```

```
        i++;
    }

    i = 0;

    while(input2[i] != '\0'){
        int c = input2[i];
        freq[c] = freq[c] - 1;
        if(freq[c] < 0){

            return 0;
        }

        i++;
    }

     return 1;
}
```

11-Ass : Remove Consecutive Duplicates

Send Feedback

For a given string(str), remove all the consecutive duplicate characters.

Example:

Input String: "aaaa"
Expected Output: "a"
Input String: "aabbbcc"
Expected Output: "abc"

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

Output Format:

The only line of output prints the updated string.

Note:

You are not required to print anything. It has already been taken care of.

Constraints:

0 <= N <= 10^6
Where N is the length of the input string.
Time Limit: 1 second

Sample Input 1:

aabccbaa

Sample Output 1:

abcba

Sample Input 2:

xxyyzxx

Sample Output 2:

xyzx

```
1.  void removeConsecutiveDuplicates(char input[]) {
2.      // Write your code here
3.      char lastchar = input[0];
4.      int i = 1, j = 1;
5.
6.      while(input[i] != '\0'){
7.
8.          if(input[i] == lastchar){
9.              i++;
10.             continue;
11.         }
12.         else{
13.         input[j] = input[i];
14.         lastchar = input[i];
15.         j++;
16.         i++;
17.         }
18.
19.     }
20.     input[j] = '\0';
21. }
```

12-Ass : Reverse Each Word

Send Feedback

Aadil has been provided with a sentence in the form of a string as a function parameter. The task is to implement a function so as to print the sentence such that each word in the sentence is reversed.

Example:

Input Sentence: "Hello, I am Aadil!"
The expected output will print, ",olleH I ma !lidaA".

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. The input string represents the sentence given to Aadil.

Output Format:

The only line of output prints the sentence(string) such that each word in the sentence is reversed.

Constraints:

0 <= N <= 10^6
Where N is the length of the input string.
Time Limit: 1 second

Sample Input 1:

Welcome to Coding Ninjas

Sample Output 1:

emocleW ot gnidoC sajniN

Sample Input 2:

Always indent your code

Sample Output 2:

```
syawlA tnedni ruoy edoc
```

```
1.  void reverseEachWord(char input[]) {
2.      // Write your code here
3.      int s = 0;
4.      int i = 0;
5.
6.      while(input[i] != '\0'){
7.
8.                  if(input[i] == ' '){
9.                      int e = i-1;
10.                             while(s < e){
11.                                 char temp = input[s];
12.                                 input[s] = input[e];
13.                                 input[e] = temp;
14.                                 s++;
15.                                 e--;
16.                             }
17.                     s = i+1;
18.
19.                 }
20.
21.         if(input[i+1] == '\0'){
22.                         int e = i;
23.                             while(s < e){
24.                                 char temp = input[s];
25.                                 input[s] = input[e];
26.                                 input[e] = temp;
27.                                 s++;
28.                                 e--;
29.                              }
30.
31.
32.         }
33.
34.
35.
36.         i++;
37.     }
38.
39. }
```

13-Ass : Remove character

Send Feedback

For a given a string(str) and a character X, write a function to remove all the occurrences of X from the given string.

The input string will remain unchanged if the given character(X) doesn't exist in the input string.

Input Format:

The first line of input contains a string without any leading and trailing spaces.
The second line of input contains a character(X) without any leading and trailing spaces.

Output Format:

The only line of output prints the updated string.

Note:

You are not required to print anything explicitly. It has already been taken care of.

Constraints:

0 <= N <= 10^6
Where N is the length of the input string.
Time Limit: 1 second

Sample Input 1:

aabccbaa
a

Sample Output 1:

bccb

Sample Input 2:

xxyyzxx
y

Sample Output 2:

xxzxx

```
1.  void removeAllOccurrencesOfChar(char input[], char c) {
2.      // Write your code here
3.      int i = 0;
4.
5.      while(input[i] != '\0'){
6.
7.          if(input[i] == c){
8.              int k = i;
9.
10.             while( input[k] != '\0'){
11.                 input[k] = input[k+1];
12.                 k++;
13.             }
14.             continue;
15.         }
16.
17.
18.         i++;
19.     }
20. }
```

14-Ass : Highest Occuring Character

Send Feedback

For a given a string(str), find and return the highest occurring character.

Example:

Input String: "abcdeapapqarr"
Expected Output: 'a'
Since 'a' has appeared four times in the string which happens to be the highest frequency character, the answer would be 'a'.
If there are two characters in the input string with the same frequency, return the character which comes first.

Consider:

Assume all the characters in the given string to be in lowercase always.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces.

Output Format:

The only line of output prints the updated string.

Note:

You are not required to print anything explicitly. It has already been taken care of.

Constraints:

0 <= N <= 10^6
Where N is the length of the input string.
Time Limit: 1 second

Sample Input 1:

abdefgbabfba

Sample Output 1:

b

Sample Input 2:

xy

Sample Output 2:

x

```
char highestOccurringChar(char input[]) {
    // Write your code here
    int freq[256] = {0,0};

    int i = 0;
    while( input[i] != '\0'){
        int c = input[i];
        freq[c] = freq[c] + 1;
        i++;
    }

    int index = 97;
    int max = freq[97];
```

```
14.
15.     for(int i = 97; i <= 122 ; i++){
16.
17.         if(freq[i] > max){
18.             max = freq[i];
19.                 index = i;
20.         }
21.     }
22.
23.     char c = index;
24.     return c;
25. }
```

15-Ass : Compress the String

Send Feedback

Write a program to do basic string compression. For a character which is consecutively repeated more than once, replace consecutive duplicate occurrences with the count of repetitions.

Example:

If a string has 'x' repeated 5 times, replace this "xxxxx" with "x5".
The string is compressed only when the repeated character count is more than 1.

Note:

Consecutive count of every character in the input string is less than or equal to 9.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces.

Output Format:

The output contains the string after compression printed in single line.

Note:

You are not required to print anything. It has already been taken care of. Just implement the given function.

Constraints:

0 <= N <= 10^6
Where 'N' is the length of the input string.
Time Limit: 1 sec

Sample Input 1:

aaabbccdsa

Sample Output 1:

a3b2c2dsa

Explanation for Sample Output 1:

In the given string 'a' is repeated 3 times, 'b' is repeated 2 times, 'c' is repeated 2 times and 'd', 's' and 'a' and occuring 1 time hence no compression for last 3 characters.

Sample Input 2:

aaabbcddeeeee

Sample Output 2:

a3b2cd2e5

Explanation for Sample Output 2:

In the given string 'a' is repeated 3 times, 'b' is repeated 2 times, 'c' is occuring single time, 'd' is repeating 2 times and 'e' is repeating 25times.

```
#include <bits/stdc++.h>
string getCompressedString(string &input) {
    // Write your code here.
    string newstr = "";
    int n = input.length();

    for (int i = 0; i < n; i++){
        int count = 1;
        char c = input[i];

        while (i < n - 1 && input[i] == input[i + 1]) {
            count++;
            i++;
        }

        if(count > 1){
            newstr +=  c + to_string(count);
        }

        if(count == 1){
            newstr += c;
        }


    }

    return newstr;
}
```