

CPP FINAL ASSIGNMENT

/*

Q3: You have a binary string with length N. You are allowed to do swap() operations of two elements only if the index parity of both elements is the same. This means that: you can swap() an element at index 2, with any element at index 4, 6, 8 etc. (even parity), similarly the element at index 3 can be swapped with the element at index 1, 5, 7, 9 etc.

Ex: Given string 1110, you can swap 2nd and 4th char to get 1011. But, you can never get 1101 if you follow the swap rules.

Now, the task is to find the number of times substring 01 can occur for all such possible swaps.

Ex: 1110 does not have any substring of 01. Now you can swap 2nd and 4th to get 1011, this will have 'one' occurrence of 01. We can now perform more swaps but we will never get any more 01 substrings here.

Input: 00100 output should be 1, for: 01010->2, for 10001->2

*/

```
#include<iostream>
```

```
#include<string>
```

```
using namespace std;
```

```
int main(){
```

```
    string str;
```

```
    cin>>str;
```

```
    int len;
```

```
    len=str.length();
```

```
    int myarray[len];
```

```
    int mynewarr[len];
```

```
    int counter=0;
```

```

for (int i=0;i<len;i++){
    //convertinf my string in array by typeconversion &ascii values.
    myarray[i] =((int) str[i]) - 48;
}
for (int i=0;i<len;i++){
    mynewarr[i] =myarray[i];
}

// for(int i=0;i<len;i++){
//   cout<<myarray[i]<<"\t";
// }
// cout<<endl;
// for(int i=0;i<len;i++){
//   cout<<mynewarr[i]<<"\t";
// }

cout<<endl;
for(int i=0;i<len;i++){
    if(i % 2 == 0){
        if(myarray[i] != myarray[i+1] ){
            int temp;
            temp = myarray[i];
            myarray[i] = myarray[i+2];
            myarray[i+2] = temp;
        }
        if((myarray[i] == 0) && (myarray[i+1] == 1) && (myarray != mynewarr)){
            counter++;
        }
    }
}

```

```

else if(i % 2 != 0){

    if(myarray[i] != myarray[i+1] ){

        int temp;

        temp = myarray[i];

        myarray[i] = myarray[i+2];

        myarray[i+2] = temp;

    }

    if((myarray[i] == 0) && (myarray[i+1] == 1) && (myarray != mynewarr)){

        counter++;

    }

}

}

cout<<counter<<endl;

}

```

The screenshot shows a Visual Studio Code editor with a C++ file named `set_3_Q3.cpp` open. The code is a C++ program that takes a string input and processes it based on certain rules. The program includes headers for `iostream` and `string`, and uses the `std` namespace. The `main` function reads a string `str` from the user, converts it to an integer array `myarray`, and then processes it based on the rules defined in the code. The program also includes a `mynewarr` array and a `counter` variable to track the number of swaps.

The terminal window shows the execution of the program. The user enters the string `1110`, and the program outputs `01010`. The user then enters `10001`, and the program outputs `00100`. The user then enters `00100`, and the program outputs `00100`.

```
/*
```

Q4: In this question, you are given a binary string of length T. Now you need to create two permutations of this string: S1 and S2 such that the 'longest common subsequence' between the two newly created strings is smallest.

Ex: Given string: 101, you can find S1: 110 and S2: 011, The longest common subsequence between S1 and S2 is 11 and the length is 2. There cannot be any two permutations of the given string where the LCS is less than 2

Ex: Given 0111, then S1 should be: 1101, and S2: 0111, the smallest LCS will be 2 char long.

```
*/
```

```
#include <string.h>
```

```
#include <iostream>
```

```
using namespace std;
```

```
int j = 0, r = 0;
```

```
int listOfRotation[] = {};
```

```
string myarray[] = {};
```

```
bool chkforswap(char str[], int beg, int curr){
```

```
    for (int i = beg; i < curr; i++)
```

```
        if (str[i] == str[curr])
```

```
            return 0;
```

```
    return 1;
```

```
}
```

```
void smallestValue(int myarray[], int N){
```

```
    int min = myarray[0];
```

```
    for (int i = 1; i < N; i++) {
```

```
        if (myarray[i] < min)
```

```
            min = myarray[i];
```

```
}
```

```
    cout << "The smallest LCS is: " << min << endl;
}
```

```
void rotatingString(char str[], int ind, int len){
    if (ind >= len){
        myarray[j] = str;
        j++;
        return;
    }
    for (int i = ind; i < len; i++){
        bool check = chkforswap(str, ind, i);
        if (check){
            swap(str[ind], str[i]);
            rotatingString(str, ind + 1, len);
            swap(str[ind], str[i]);
        }
    }
}
```

```
int LCS(string X, string Y, int m, int n){
```

```
    int LCSuff[m + 1][n + 1];
```

```
    int result = 0;
```

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

```
        for (int j = 0; j <= n; j++){
```

```
            if (i == 0 || j == 0)
```

```
                LCSuff[i][j] = 0;
```

```

        else if (X[i - 1] == Y[j - 1]){
            LCSuff[i][j] = LCSuff[i - 1][j - 1] + 1;
            result = max(result, LCSuff[i][j]);
        }
        else
            LCSuff[i][j] = 0;
    }
}
listOfRotation[r] = result;
r++;
return result;
}

```

```

int main(){
    char string[20];
    int len;
    cout << "Enter string : ";
    cin >> string;

    len = strlen(string);
    rotatingString(string, 0, len);
    if (j == 2)
        cout << "The smallest LCS is: 0" << endl;
    else{
        for (int i = 0; i < j; i++){
            for (int k = i + 1; k < j; k++){
                LCS(myarray[i], myarray[k], len, len);
            }
        }
    }
}

```

```

}

smallestValue(listOfRotation, r);

return 0;

}

```

The screenshot shows the Visual Studio Code editor with a C++ file named `set_3_Q4.cpp`. The code implements a function `smallestValue` to find the minimum length of two strings `S1` and `S2` such that their concatenation is a permutation of a given string `str`. The code uses a recursive helper function `chkforswap` to check if two strings are permutations of each other.

```

1  /*
2  Q4: In this question, you are given a binary string of length T. Now you need to create two
3  permutations of this string: S1 and S2 such that the 'longest common subsequence' between the two
4  newly created strings is smallest.
5  Ex: Given string: 101, you can find S1: 110 and S2: 011. The longest common subsequence between
6  S1 and S2 is 11 and the length is 2. There cannot be any two permutations of the given string where
7  the LCS is less than 2
8  Ex: Given 0111, then S1 should be: 1101, and S2: 0111, the smallest LCS will be 2 char long
9  */
10 #include <string.h>
11 #include <iostream>
12 using namespace std;
13 int j = 0, r = 0;
14 int listOfRotation[] = {};
15 string myarray[] = {};
16
17 bool chkforswap(char str[], int beg, int curr){
18     for (int i = beg; i < curr; i++){
19         if (str[i] == str[curr])
20             return 0;
21     }
22     return 1;
23 }
24
25 void smallestValue(int myarray[], int N){
26     int min = myarray[0];
27     for (int i = 1; i < N; i++){
28         if (myarray[i] < min)
29             min = myarray[i];
30     }
31     cout << "The smallest LCS is: " << min << endl;
32 }
33
34 void rotationString(char str[], int ind, int len){

```

The terminal window shows the execution of the program:

```

nirmal@nirmal-VirtualBox: ~/user/cpp/finalAssignment
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ g++ set_3_Q4.cpp
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ ./a.out
Enter string : 101
The smallest LCS is: 2
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ ./a.out
Enter string : 0111
The smallest LCS is: 2
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$

```

/*There are two processes, A and B, that can access a common variable x. They can access it in sequence, A first and then B, or B first then A. But in one day they access it only once. A logbook is maintained by the OS showing which process accessed x when.

You got a series of entries of the log, but you want to make sure that this log has not been altered by anyone. Your output is the answer to the question: Is the log valid or not?

Ex: Input: AB, output: Yes

Input: ABAABB, output: No*/

```
#include<iostream>
```

```
#include<string>
```

```
using namespace std;
```

```
int main(){
```

```
    string x;
```

```
    cout<<"Enter log string:"<<endl;
```

```
    cin>>x;
```

```
    bool ans;
```

```
    int len;
```

```
    char leta='A',letb='B';
```

```
    len = x.length();
```

```
    if(x[0] == leta){
```

```
        for (int i=0;i<len;i+=2){
```

```
            if((x[i] == leta) && (x[i+1] == letb)){
```

```
                ans = true;
```

```
            }
```

```
        else{
```

```
            ans = false;
```



```

    }
}
}

if(x[0] == letb){
    for (int i=0;i<len;i+=2){
        if((x[i] == letb) && (x[i+1] == leta)){
            ans = true;
        }
        else{
            ans = false;
        }
    }
}

if(ans)
    cout<<"YES"<<endl;
else
    cout<<"NO"<<endl;
}

```

The screenshot shows the Visual Studio Code interface with a C++ file named `set_3_Q8.cpp` open. The code in the editor is as follows:

```

1  /*There are two processes, A and B, that can access a common variable x. They can access it in
2  sequence, A first and then B, or B first then A. But in one day they access it only once. A logbook is
3  maintained by the OS showing which process accessed x when.
4
5  You got a series of entries of the log, but you want to make sure that this log has not been
6  altered by anyone. Your output is the answer to the question: Is the log valid or not?
7  Ex: Input: AB, output: Yes
8  Input: ABAAAB, output: No*/
9  #include<iostream>
10 #include<string>
11 using namespace std;
12
13 int main(){
14     string x;
15
16     cout<<"Enter log string:"<<endl;
17     cin>>x;
18     bool ans;
19
20     int len;
21     char leta='A',letb='B';
22     len = x.length();
23     if(x[0] == leta){
24         for (int i=0;i<len;i+=2){
25             if((x[i] == leta) && (x[i+1] == letb)){
26                 ans = true;
27             }
28             else{
29                 ans = false;
30             }
31         }
32     }
33     if(x[0] == letb){

```

Below the code editor, a terminal window is open, showing the execution of the program. The user enters the log string "AB" and "ABAAAB", and the program outputs "YES" and "NO" respectively.

```

nirmal@nirmal-VirtualBox: ~/user/cpp/finalAssignment
File Edit View Search Terminal Help
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ g++ set_3_Q8.cpp
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ ./a.out
Enter log string:
AB
YES
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ ./a.out
Enter log string:
ABAAAB
NO
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$

```