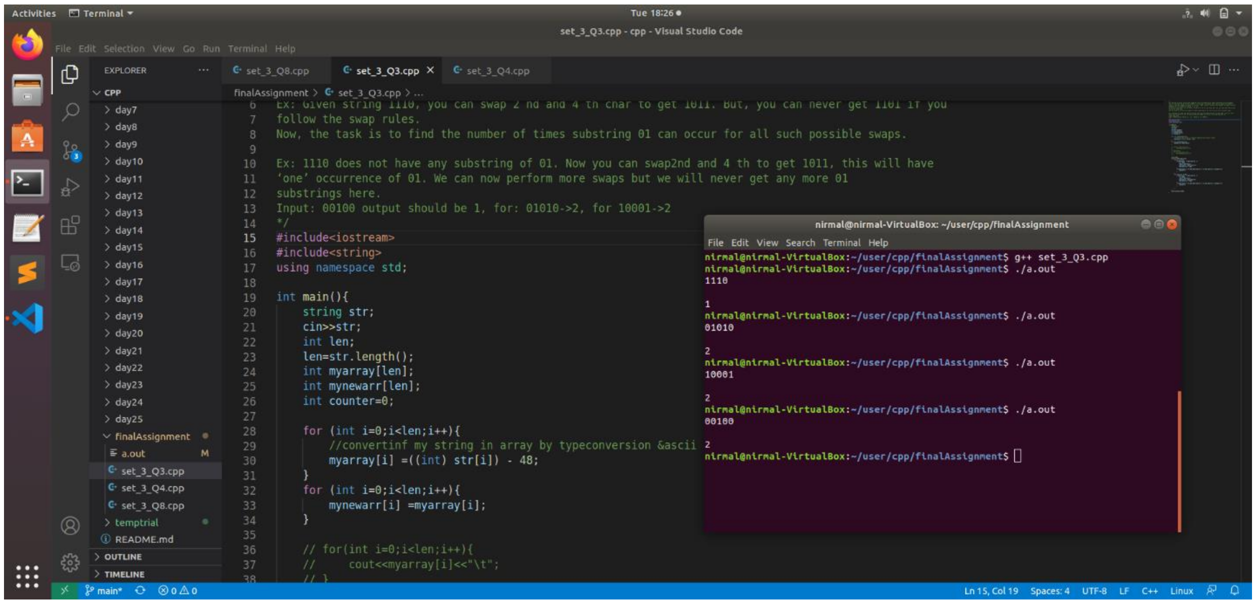
CPP FINAL ASSIGNMENT

```
/*
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 index1, 5,7,9 etc.
Ex: Given string 1110, you can swap 2 nd and 4 th 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 swap2nd and 4 th 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;
}
```

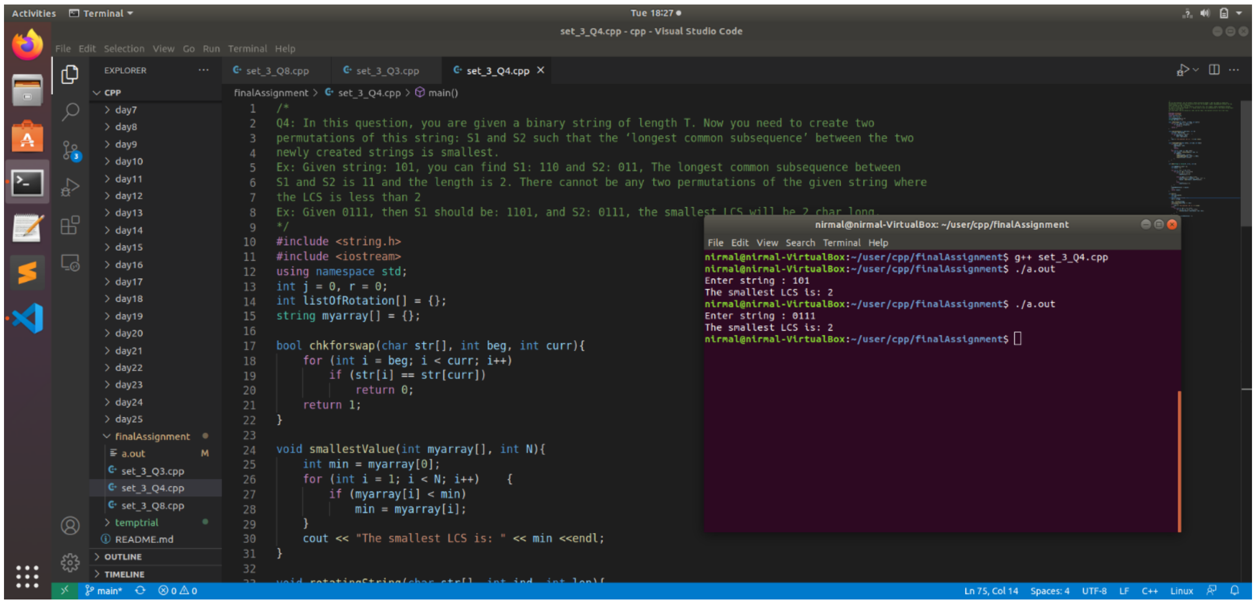


```
Ex: Given string 1110, you can swap 2 nd and 4 th char to get 1011. But, you can never get 1101 if you
follow the swap rules.
Now, the task is to find the number of times substring 01 can occur for all such possible swaps.

Ex: 1110 does not have any substring of 01. Now you can swap2nd and 4 th 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
        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";
    // }
```

```
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ g++ set_3_Q3.cpp
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ ./a.out
1110

1
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ ./a.out
01010

2
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ ./a.out
10001

2
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ ./a.out
00100

2
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ 
```

```
/*
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;
}
```

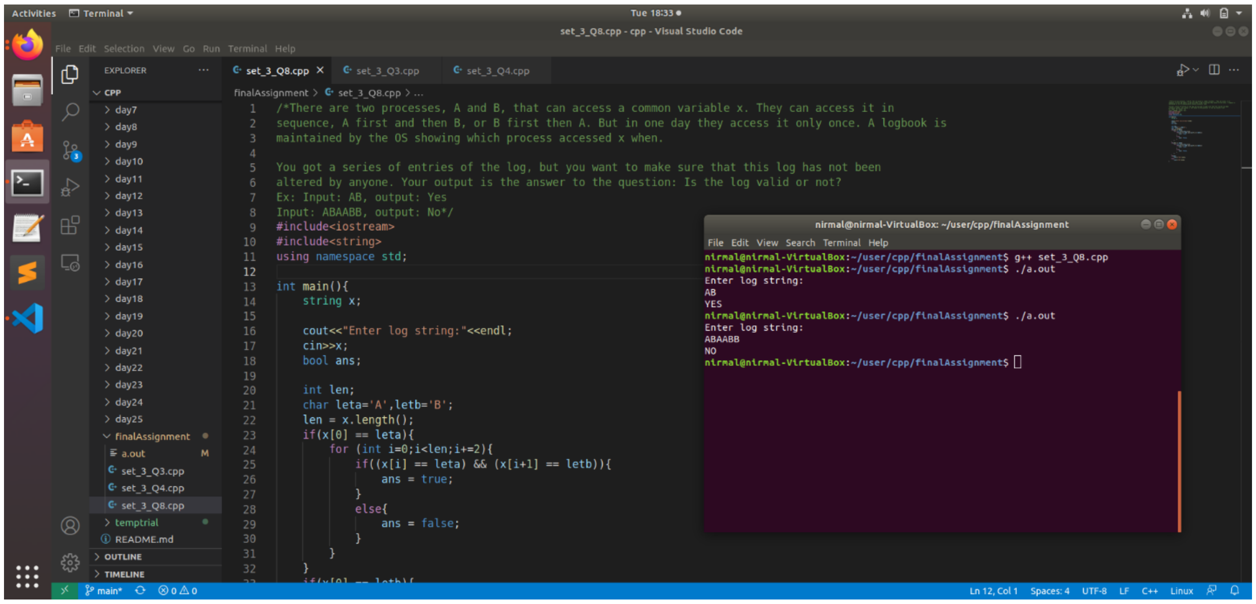


```cpp
/*
Q4: In this question, you are given a binary string of length T. Now you need to create two
permutations of this string: S1 and S2 such that the ‘longest common subsequence’ between the two
newly created strings is smallest.
Ex: Given string: 101, you can find S1: 110 and S2: 011, The longest 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){
```

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;
}
```



```
/*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;
            }
        }
    }
```

```
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:
ABAABB
NO
nirmal@nirmal-VirtualBox:~/user/cpp/finalAssignment$ 
```