Minimum number of bracket reversals needed to make an expression balanced

Given an expression with only '}' and '{'. The expression may not be balanced. Find minimum number of bracket reversals to make the expression balanced.

Examples:

```
Input:  exp = "}{"
Output: 2
We need to change '}' to '{' and '{' to
'}' so that the expression becomes balanced,
the balanced expression is '{}'

Input:  exp = "{{{"
Output: Can't be made balanced using reversals

Input:  exp = "{{{{"
Output: 2

Input:  exp = "{{{{}}"
Output: 1

Input:  exp = "}{{}}{{{"
Output: 3
```

We strongly recommend you to minimize your browser and try this yourself first.

One simple observation is, the string can be balanced only if total number of brackets is even (there must be equal no of '{' and '}')

A **Naive Solution** is to consider every bracket and recursively count number of reversals by taking two cases (i) keeping the bracket as it is (ii) reversing the bracket. If we get a balanced expression, we update result if number of steps followed for reaching here is smaller than the minimum so far. Time complexity of this solution is $O(2^n)$.

An **Efficient Solution** can solve this problem in O(n) time. The idea is to first remove all balanced part of expression. For example, convert "}**{{}}**{{{" to "}{{{" by removing highlighted part. If we take a closer look, we can notice that, after removing balanced part, we always end up with an expression of the form }}…}{{…{, an expression that contains 0 or more number of closing brackets followed by 0 or more numbers of opening brackets.

How many minimum reversals are required for an expression of the form "}}..}{{..{" ?. Let m be the total number of closing brackets and n be the number of opening brackets. We need $\lceil m/2 \rceil + \lceil n/2 \rceil$ reversals. For example }}}}{{ requires 2+1 reversals.

Below is C++ implementation of above idea.

```cpp
// C++ program to find minimum number of
// reversals required to balance an expression
#include<bits/stdc++.h>
using namespace std;


// Returns count of minimum reversals for making
// expr balanced. Returns -1 if expr cannot be
// balanced.
int countMinReversals(string expr)
{
    int len = expr.length();

    // length of expression must be even to make
    // it balanced by using reversals.
    if (len%2)
       return -1;

    // After this loop, stack contains unbalanced
    // part of expression, i.e., expression of the
    // form "}}..}{{..{"
    stack<char> s;
    for (int i=0; i<len; i++)
    {
        if (expr[i]=='}' && !s.empty())
        {
            if (s.top()=='{')
                s.pop();
            else
                s.push(expr[i]);
        }
        else
            s.push(expr[i]);
    }

    // Length of the reduced expression
    // red_len = (m+n)
    int red_len = s.size();

    // count opening brackets at the end of
    // stack
    int n = 0;
    while (!s.empty() && s.top() == '{')
    {
        s.pop();
        n++;
    }

    // return ceil(m/2) + ceil(n/2) which is
    // actually equal to (m+n)/2 + n%2 when
    // m+n is even.
    return (red_len/2 + n%2);
}

// Driver program to test above function
int main()
{
   string expr = "}}{{";
   cout << countMinReversals(expr);
   return 0;
}
```

Output:

```
2
```

Time Complexity: O(n)

Auxiliary Space: O(n)