Minimum characters to be added at front to make string palindrome

Given string **str** we need to tell minimum characters to be added in front of the string to make string palindrome.

Examples:

```
Input  : str = "ABC"
Output : 2
We can make above string palindrome as "CBABC"
by adding 'B' and 'C' at front.
Input  : str = "AACECAAAA";
Output : 2
We can make above string palindrome as AAAACECAAAA
by adding two A's at front of string.
```

Naive approach: Start checking the string each time if it is a palindrome and if not, then delete the last character and check again. When the string gets reduced to either a palindrome or an empty string then the number of characters deleted from the end till now will be the answer as those characters could have been inserted at the beginning of the original string in the order which will make the string a palindrome.

Below is the implementation of the above approach:

```
// C++ program for getting minimum character to be

// added at front to make string palindrome

#include<bits/stdc++.h>

using namespace std;

// function for checking string is palindrome or not

bool ispalindrome(string s)

{

        int l = s.length();

        int j;


        for(int i = 0, j = l - 1; i <= j; i++, j--)

        {

                if(s[i] != s[j])
```

```
                return false;
        }
        return true;
}

// Driver code
int main()
{
        string s = "BABABAA";
        int cnt = 0;
        int flag = 0;

        while(s.length()>0)
        {
                // if string becomes palindrome then break
                if(ispalindrome(s))
                {
                        flag = 1;
                        break;
                }
                else
                {
                cnt++;

                // erase the last element of the string
                s.erase(s.begin() + s.length() - 1);
                }
        }
```

```
        // print the number of insertion at front
        if(flag)
                cout << cnt;
}
```

Output
2

Time complexity: O(n2)
Auxiliary Space: O(1)

Another approach using **“Two Pointers”:-**

- Initialize two pointers start and end to the beginning and end of the string, respectively.
- While start is less than end, if the characters at the start and end pointers are equal, move the start pointer one position to the right and the end pointer one position to the left. If the characters are not equal, increment the res variable (which keeps track of the number of characters that need to be added) and reset the start and end pointers to the beginning and end of the string with a reduced number of characters.
- When start is no longer less than end, return the value of res as the minimum number of characters that need to be added to the front of the string to make it a palindrome.

```
#include <iostream>
#include <string>

using namespace std;

class Solution {
public:
        int addMinChar(string str1) {
```

```
        int n = str1.length();
        int start = 0;
        int end = n - 1;
        int res = 0;
        while (start < end) { // While the pointers have not met in the middle of the string
            if (str1[start] == str1[end]) { // If the characters at the start and end pointers are equal
                start++; // Move the start pointer to the right
                end--; // Move the end pointer to the left
            }
            else {
                res++; // Increment the count of characters to be added
                start = 0; // Reset the start pointer to the beginning of the string
                end = n - res - 1; // Reset the end pointer to the end of the string with a reduced number of characters
            }
        }
        return res; // Return the count of characters to be added
    }
};

int main() {
    Solution sol;
    string str = "AACECAAAA";
    cout << sol.addMinChar(str) << endl;
    return 0;
}
```