Char Array and String - 02

Q. Remove all adjacent duplicate in a string :

LeetCode Solution:

```
class Solution {
public:
    string removeDuplicates(string s) {
        string ans = "";
        int n=s.length();
        for(int i=0; i<n; i++){
            char currChar = s[i];
            if(ans.empty()){
                ans.push_back(currChar);
            }
            else if(currChar==ans.back()){
                ans.pop_back();
            }
            else if(currChar != ans.back()){
                ans.push_back(currChar);
            }
        }
        return ans;
    }
};
```

Q. Remove all adjacent duplicate in a string-II :

LeetCode Solution:

```
// baki
```

Q. Remove All Occurrences of a Substring :

LeetCode Solution:

```
class Solution {
public:
    string removeOccurrences(string full, string pattern) {
        while(full.find(pattern)!=string::npos){
            full.erase(full.find(pattern), pattern.length());
        }
    return full;
```

```
    }
};
```

Q. Valid Palindrome-II :

LeetCode Solution:

```
class Solution {
public:
    bool checkpalindrome(string str, int s, int e){
        while(s<=e){
            if(str[s]!=str[e]){
                return false;
            }
            else{
                s++;
                e--;
            }
        }
        return true;
    }

    bool validPalindrome(string s) {
        int i=0;
        int j=s.length()-1;

        while(i<=j){
            if(s[i]==s[j]){
                i++;
                j--;
            }
            else{
                bool ansOne = checkpalindrome(s,i+1,j);
                bool ansTwo = checkpalindrome(s,i,j-1);
                bool ans = ansOne||ansTwo;
                return ans;
            }
        }
        return true;
    }
};
```

Q. Minimum Time Difference (539)

LeetCode Solution:

```
// baki
```

Q. Palindromic Substrings (647)

LeetCode Solution:

```
class Solution {
public:
    int expandAroundCenter(string str, int i, int j){
        int cnt=0;
        while(i>=0 && j<str.length() && str[i] == str[j]){
            cnt++;
            i--;
            j++;
        }
        return cnt;
    }

    int countSubstrings(string s) {
        int totalCnt = 0;
        for(int center=0; center<s.length(); center++){
            // odd
            int i=center;
            int j=center;
            int oddPalSubStringKaCnt = expandAroundCenter(s,i,j);

            // even
            i=center;
            j=center+1;
            int evenPalSubStringKaCnt = expandAroundCenter(s,i,j);
            totalCnt = totalCnt + oddPalSubStringKaCnt + evenPalSubStringKaCnt;

        }
        return totalCnt;
    }
};
```