```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

// To remove outer parantheses.

string removeOuterParentheses (string s) {
    int cnt =0;
    string t = " ";
    for (int i=0; i<s.length(); i++) {
        if (s[i] == ')') {
            cnt --;
        }
        if (cnt !=0) {
            t.push_back (s[i]);
        }
        if (s[i] == '(') {
            cnt++;
        }
    }
    return t;
}

// To reverse word

string reverseWord (string s1) {
 string res;
 int i=0;
 int n = s1.length();
```

VISION

```
while (i < n) {
    while (i < n && s1[i] == ' ') {
        i++;
    }
    if (i >= n) {
        break;
    }
    int j = i + 1;
    while (j < n && s1[j] != ' ') {
        j++;
    }
    string sub = s1.substr(i, j - i);
    if (res.length() == 0) {
        res = sub;
    } else {
        res = sub + " " + res;
    }
    i = j + 1;
}
return res;
}

// To find largest odd number in a string

String largestOddNumber (string num) {
    for (int i = num.length() - 1; i >= 0; i--) {
        if ((num[i] - '0') % 2 != 0) {
            return num.substr(0, i + 1);
        }
    }
    return "";
}
```

```
// To check if the string is Isomorphic

bool isIsomorphic (string s2, string t) {
        char n[128] = {};
        for (int i = 0; i < s2.length(); i++){
            char c = s2[i];
            if (!n[c]) {
                for (char s2 : n){
                    if (s2 == t[i]) {
                        return false;
                    }
                }
                n[c] = t[i];
            } else if (n[c] != t[i]) {
                return false;
            }
        }
        return true;
    }

// To convert Roman to integer

int romanToInt (string s4) {
        map <char, int> rom;
        rom.insert (make_pair ('I', 1));
        rom.insert (make_pair ('V', 5));
        rom.insert (make_pair ('X', 10));
        rom.insert (make_pair ('L', 50));
        rom.insert (make_pair ('C', 100));
        rom.insert (make_pair ('D', 500));
        rom.insert (make_pair ('M', 1000));
```

```
int n = s4.length(), num, sum=0;
for (int i=0; i<n; ){
    if (i==(n-1) || (rom[s4[i]] >= rom[s4[i+1]]))
    {
        num = rom[s4[i]];
        i++;
    } else {
        num = rom[s4[i+1]] - rom[s4[i]];
        i+2;
    }
    sum += num;
}
return sum;
}

// to find substring

long long upto(string s3, int n){
    int l=0, r=0, cnt=0;
    long long res=0;

    vector<int> mp(26,0);
    while(r<s3.length()){
        mp[s3[r]-'a'] ~~++~~;
        ~~cnt++~~;
        if(mp[s3[r]-'a']==1){
            cnt++;
        }
        while(cnt>n){
            mp[s3[l]-'a']--;
            if(mp[s3[l]-'a']<=0){
```

```
            cnt--;
        }
        l++;
    }
    res += r-l+1;
    r++;
}
return res;
}
long long int subStrCount (string s3, int k) {
    return upto(s3,k) - upto(s3,k-1);
}
```