Rock Paper Scissor :

Chef and Chefina are playing the famous game of Rock, Paper, Scissors.
প্রতি রাউন্ডে যদি দুইজন একই move দেয় তাহলে সেটা draw হয়।
Rock জেতে Scissors এর বিরুদ্ধে, Scissors জেতে Paper এর বিরুদ্ধে, আর Paper জেতে Rock এর বিরুদ্ধে।
গেম শেষে যার পয়েন্ট বেশি, সেই জয়ী।

এখন আমাদের বের করতে হবে Chef–এর ন্যূনতম কয়টা রাউন্ড পরিবর্তন করতে হবে যাতে সে গেম জেতে।

প্রথমে আমরা গুনে নিই কয়টা রাউন্ডে Chef জিতেছে **(`counter1`)** এবং কয়টা রাউন্ডে Chefina জিতেছে **(`counter2`)।**
যদি Chef আগেই বেশি রাউন্ড জেতে, তাহলে কোনো পরিবর্তনের দরকার নেই।

অন্যথায়, Chef কে এমনভাবে কিছু রাউন্ড পরিবর্তন করতে হবে যাতে তার মোট জয় Chefina–এর চেয়ে একটি বেশি হয়।
তাহলে প্রয়োজনীয় পরিবর্তন হবে -

((counter1 + counter2) / 2) + 1 - counter1

এই মানটাই Chef–এর ন্যূনতম পরিবর্তনের সংখ্যা।

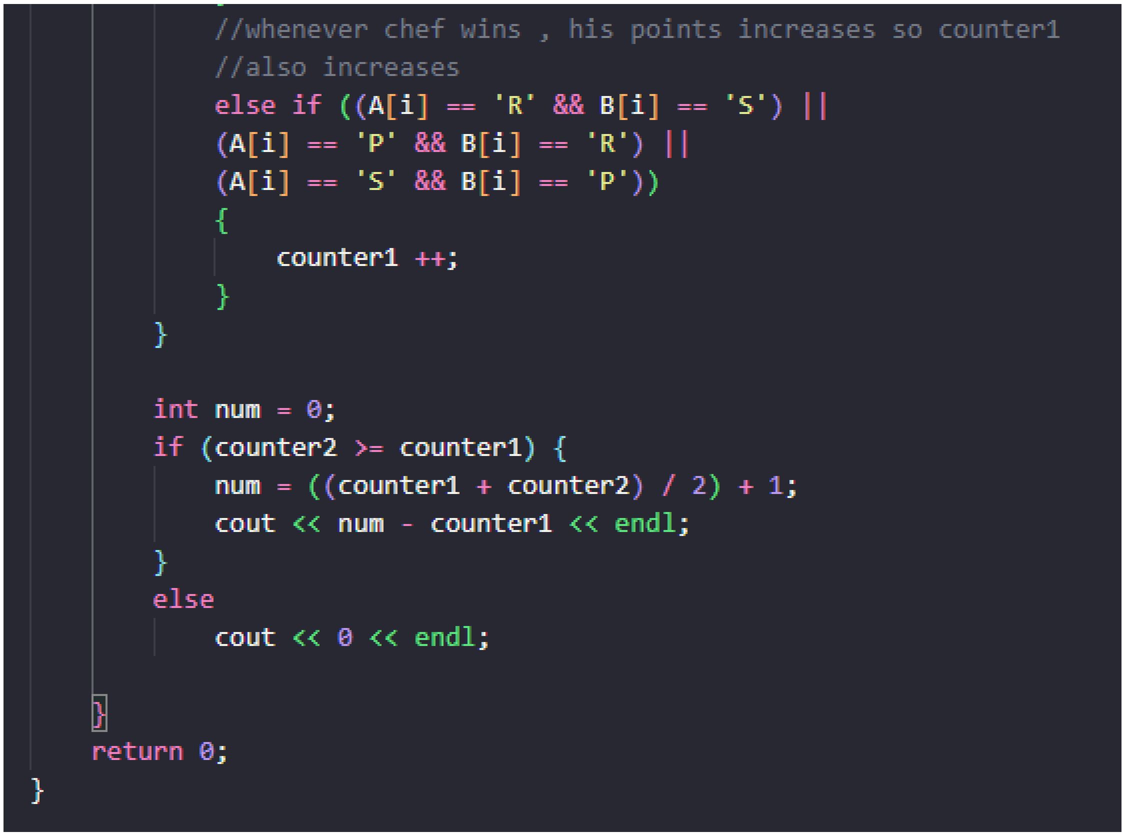
`(counter1 + counter2) / 2` মানে দুইজনের সমান অবস্থার পয়েন্ট।

জিততে হলে Chef–কে তার থেকে ১ বেশি পেতে হবে, তাই +1।

তার বর্তমান জয় `counter1`, তাই প্রয়োজনীয় পরিবর্তন =

লক্ষ্য জয় **–** বর্তমান জয় **= ((counter1 + counter2) / 2) + 1 - counter1**.

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

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        string A ;
        cin >> A ;
        string B ;
        cin>> B;

        int counter1 = 0, counter2 = 0;

        for (int i = 0; i < n; i++) {

            //whenever chefina wins , her points increases so counter2
            //also increases
            if ((A[i] == 'R' && B[i] == 'P') ||
            (A[i] == 'S' && B[i] == 'R') ||
            (A[i] == 'P' && B[i] == 'S'))
            {
                counter2 ++;
            }
            //whenever chef wins , his points increases so counter1
```



```
            //whenever chef wins , his points increases so counter1
            //also increases
            else if ((A[i] == 'R' && B[i] == 'S') ||
            (A[i] == 'P' && B[i] == 'R') ||
            (A[i] == 'S' && B[i] == 'P'))
            {
                counter1 ++;
            }
        }

        int num = 0;
        if (counter2 >= counter1) {
            num = ((counter1 + counter2) / 2) + 1;
            cout << num - counter1 << endl;
        }
        else
            cout << 0 << endl;

    }
    return 0;
}
```

Binary Conversion :

আমাদেরকে দুটি binary string S এবং T এবং একটি সংখ্যা K দেওয়া আছে।

প্রতিটি অপারেশনে আমরা দুটি আলাদা ইনডেক্স i ও j বেছে নিয়ে S[i] এবং S[j] swap করতে পারি।

আমাদের বের করতে হবে, ঠিক K বার swap করে S কে T তে রূপান্তর করা সম্ভব কি না।

সমাধান:

Swap করলে শুধুমাত্র S–এর মধ্যে 0 ও 1–এর অবস্থান পরিবর্তন হয়।

তাই প্রথম শর্ত —

S ও T তে 0 এবং 1–এর সংখ্যা সমান হতে হবে।

নইলে কখনোই S কে T বানানো সম্ভব না।

এখন আমরা গুনবো কয়টি পজিশনে S[i] != T[i] — ধরি এটি c।

একবার swap করলে দুইটি mismatch ঠিক হতে পারে।

তাই দরকারি ন্যূনতম swap সংখ্যা হবে c / 2।

যদি c / 2 == K, তাহলে perfect fit — উত্তর YES।

যদি c / 2 > K, তাহলে দরকারের চেয়ে কম swap সম্ভব — NO।

যদি c / 2 < K,

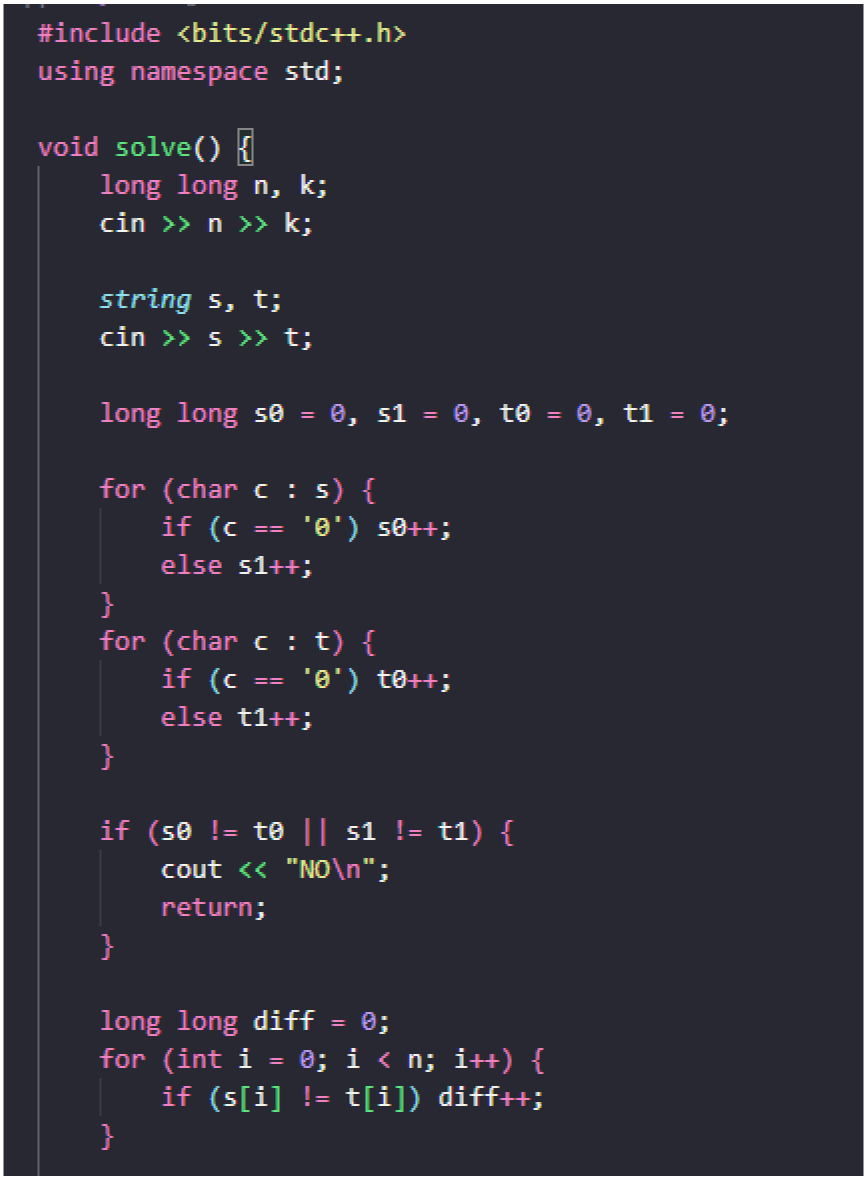
তাহলে কিছু বাড়তি swap করতে পারব, যেগুলো আবার আগের অবস্থায় ফিরিয়ে আনা সম্ভব।

এতে parity (even–odd) অনুযায়ী কিছু বিশেষ কেস তৈরি হয় —

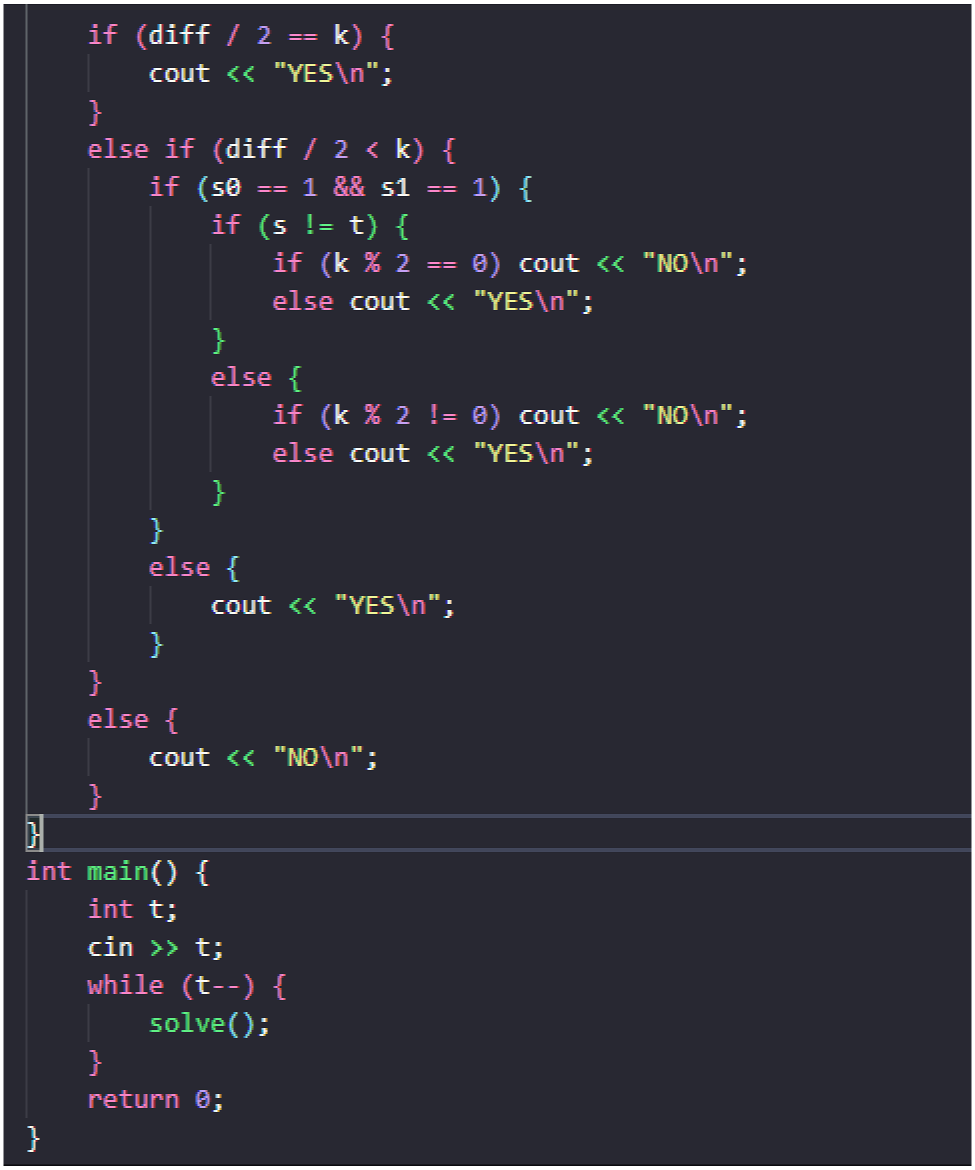
যদি S ও T তে মাত্র ১টা ‘0’ এবং ১টা ‘1’ থাকে, তখন parity অনুযায়ী check করতে হবে

যে K জোড় না বিজোড়।

অন্য ক্ষেত্রে সবসময়ই বাড়তি swap করা সম্ভব, তাই উত্তর YES।



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

void solve() {
    long long n, k;
    cin >> n >> k;

    string s, t;
    cin >> s >> t;

    long long s0 = 0, s1 = 0, t0 = 0, t1 = 0;

    for (char c : s) {
        if (c == '0') s0++;
        else s1++;
    }
    for (char c : t) {
        if (c == '0') t0++;
        else t1++;
    }

    if (s0 != t0 || s1 != t1) {
        cout << "NO\n";
        return;
    }

    long long diff = 0;
    for (int i = 0; i < n; i++) {
        if (s[i] != t[i]) diff++;
    }
```



```
    if (diff / 2 == k) {
        cout << "YES\n";
    }
    else if (diff / 2 < k) {
        if (s0 == 1 && s1 == 1) {
            if (s != t) {
                if (k % 2 == 0) cout << "NO\n";
                else cout << "YES\n";
            }
            else {
                if (k % 2 != 0) cout << "NO\n";
                else cout << "YES\n";
            }
        }
        else {
            cout << "YES\n";
        }
    }
    else {
        cout << "NO\n";
    }
}
int main() {
    int t;
    cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

Sum of Four Values

আমাদের n সাইজের একটি Array দেওয়া আছে এবং একটি সংখ্যা x দেওয়া আছে। আমাদের Array থেকে এমন চারটি সংখ্যা নিতে হবে যেন তাদের যোগফল x হয়। সম্ভব হলে যেকোনো সঠিক উত্তরের index গুলো প্রিন্ট করবো আর সম্ভব না হলে impossible প্রিন্ট করতে হবে।

এই প্রব্লেম সলভ এর পূর্বে আমরা Two Pointer এর মাধ্যমে দুইটি সংখ্যার যোগফল x হয় কি না এই প্রব্লেমটা সলভ করতে হবে। যেটা আমাদের Week 4 এর কন্টেস্টে আছে।
https://vjudge.net/contest/760651#problem/O

এখন যদি আমরা দুইটি সংখ্যার যোগফল x হয় কি না বের করতে পারি। তাহলে এই প্রব্লেমটাও সলভ করা পসিবল। চারটি সংখ্যার যোগফলকে আমরা দুইটি সংখ্যার যোগফলের সাথে আরও দুইটি সংখ্যার যোগফল যোগ করে পেতে পারি। অর্থাৎ a + b + c + d = (a + b) + (c + d)

আমরা শুরুতে আমাদের Array এর যেকোনো দুইটি সংখ্যা নিয়ে যতগুলো সংখ্যা বানানো পসিবল সবগুলো একটা vector(sums) এ স্টোর করে ফেলতে পারি $O(n^2)$ কমপ্লেক্সিটিতে। n <= 1000 হওয়ায় $O(n^2)$ এ সমস্যা নেই। তাহলে এখন প্রব্লেমটা দাঁড়ালো এই নতুন ভেক্টরের দুইটা এলিমেন্ট নিয়ে আমরা তাদের যোগফল x বানাতে পারি কি না। এটা কিন্তু সম্পূর্ণই আগের প্রব্লেম, two pointer এর মাধ্যমে করে ফেলা যাবে।

এখানে একটা সমস্যা ইনডেক্সগুলো প্রিন্ট করতে হবে, তাই আমরা sums ভেক্টরে পুশ করার সময় ইনডেক্স সহ পুশ করবো। এবং দ্বিতীয় সমস্যা sums ভেক্টরের দুইটা ইনডেক্স এর সাম হয়তো x হলো কিন্তু তারা যে ইনডেক্স থেকে তৈরি হয়েছে সেগুলোর মাঝে কমন ইনডেক্স থাকতে পারে। তাই আমরা সমান ভ্যালুর জন্য All possible চেক করতে পারি, যেটা স্যলুশনে while loop দিয়ে করা হয়েছে।

একটা উদাহরণের মাধ্যমে ভালোভাবে বোঝা যাক।
ধরলাম একটা অ্যারে আছে - 2 3 6 4 4 1. এবং target sum 15.
তাহলে sums ভেক্টর হবে - [{5,1, 2}, {8, 1, 3}, {6, 1, 4}, {6, 1, 5}, {3, 1, 6}, {9, 2, 3}, {7, 2, 4}, {7, 2, 5}, {4, 2, 6}, {10, 3, 4}, {10, 3, 5}, {7, 3, 6}, {8, 4, 5}, {5, 4, 6}, {5, 5, 6}]
এবং এটাকে sort করলে হবে - [{3, 1, 6}, {4, 2, 6}, {5, 1, 2}, {5, 4, 6}, {5, 5, 6}, {6, 1, 4}, {6, 1, 5}, {7, 2, 4}, {7, 2, 5}, {7, 3, 6}, {8, 1, 3}, {8, 4, 5}, {9, 2, 3}, {10, 3, 4}, {10, 3, 5}]

এখন আমরা যদি two pointer চালাই {5, 5, 6}, {10, 3, 5} এই দুইটা যোগ করলেও কিন্তু 15 হয় কিন্তু এখানে কমন ইনডেক্স 5 আছে তাই এটা সঠিক আন্সার না। তাই পরবর্তী সমান ভ্যালুর সবগুলো loop দিয়ে চেক করলে আমরা সঠিক আন্সার পেয়ে যাবো - {5, 5, 6}, {10, 3, 4}. অর্থাৎ 3, 4, 5, 6 তম ইনডেক্সের যোগফল 15.

```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long

int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    int n, x;
```

```
    cin >> n >> x;
    vector<int> a(n);
    for (auto &u : a)
        cin >> u;

    vector<pair<ll, pair<ll, ll>>> sums;

    for (int i = 0; i < n; i++)
    {
        for (int j = i + 1; j < n; j++)
            sums.push_back({a[i] + a[j], {i, j}});
    }

    sort(sums.begin(), sums.end());
    for (int i = 0, j = sums.size() - 1, k; i < j;)
    {
        if (sums[i].first + sums[j].first == x)
        {
            while (sums[i].first + sums[j].first == x)
            {
                k = j;
                while (sums[i].first + sums[k].first == x)
                {
                    set<int> indices;
                    indices.insert(sums[i].second.first);
                    indices.insert(sums[i].second.second);
                    indices.insert(sums[k].second.first);
                    indices.insert(sums[k].second.second);
                    if (indices.size() == 4)
                    {
                        for (auto u : indices)
                            cout << u + 1 << " ";
                        return 0;
                    }
```

```
                    k--;
                }
                i++;
            }
            j = k;
        }
        else if (sums[i].first + sums[j].first < x)
            i++;
        else
            j--;
    }

    cout << "IMPOSSIBLE\n";

    return 0;
}
```