Problem E - Flip Prefix

- **Question Analysis:**
  আমাকে একটা **binary string S** দেওয়া আছে। এই **S-** কে আমাদের নিচের **operation -** এর মাধ্যমে কতধরণের **String** বানাতে পারি তা **output** দিতে হবে।
  Operation হলোঃ
  যেকোনো **length -** এর **prefix choose** কর **S[1.....i]** , যাতে ওই **prestring** এ, **'1'** এর **count** এবং **'0'** এর **count** সমান থাকে।

- **Some Key Takeaways:**
  ধর, তোমার কাছে ২টা বক্সে , ২ টা করে বল আছে।
  Box-a: a1 , a2 Box-b: b1,b2
  কতভাবে তুমি ২টা ভিন্ন ভিন্ন বক্স থেকে ২টা করে বল **choose** করতে পারবে?
  Answer : 4.
  [{a1,b1} , {a1,b2} , {a2,b1} , {a2,b2}]
  অর্থাৎ, **count_ball_Box-a X count_ball_Box-b**

- **Observation:**
  1. আমরা শুরু থেকে যদি **0 & 1 count** করে করে যায়, তাহলে যখনই **count** সমান হবে - তখনই কিন্তু এই **prefix** কে **operation** টা চালাতে পারি।
  2. তাহলে যখনই একটা এমন **prefix** পাচ্ছি, তার ২ টা **choice** পাচ্ছি, মানে **flip** করলে একধরণের **string,** নাকরলে আরেকধরণের।
     Ex - 001011 01001
     110100 01001
  3. তাহলে এমন যত **prefix** পাব, তাদের এমন ২ টা করে **Choice.**
  4. তাহলে তো হয়েই গেলো, এমন যতো **prefix** আছে তা বের করব, সবার ২ টা করে **choice,** তাই সবগুলো গুণ করব **- Some Key Takeaways -** এর মতো।

- **Implementation:**
  Loop চালিয়ে **0 & 1 count** করব, যখনই, **count equal** হবে, **answer** এর সাথে ২ করে গুণ করতে থাকব। **answer 1** দিয়ে **intitialize** থাকবে, কারণ, আমাদের দেওয়া **String** টাও তো একটা **valid string,** তাই না!

- **Code:**

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

#define ll long long

int main(){
    int tc=1;
    cin >> tc;
    while(tc--){
        int n;
        cin >> n;
        string s;
        cin >> s;

        ll ans = 1;
        int zero=0,one=0;
        for(int i=0; i<n; i++){
            if(s[i]=='0') zero++;
            else one++;

            if(zero==one) ans *=2;
        }

        cout << ans << endl;
    }

    return 0;
}
/*   Author: Hridoy Barua (CS Instructor Phitron)   */
```

Problem F - Find Outside Array

- **Question Analysis:**
  আমাদের একটা **array - A** দেওয়া আছে। আমাকে এমন একটা **integer X** বের করতে হবে, যাতে **X array -** তে না থাকে। এখানে **$X = A_i + A_j$ $(1 \le i,j \le N)$**।
  যদি, এমন কোনো **A_i, A_j** পাওয়া যায়, যাতে **X array -** তে না থাকে, তাহলে **A_i, A_j - output** দিতে হবে।
  নাহলে **-1 output** দিতে হবে।

- **Observation:**
  1. আমি কিন্তু একই **element -** ই ২ বার **choose** করতে পারি, যেহেতু **i != j** এমন কিছু বলা নাই।
  2. যদি আমরা সবথেকে বড় **element** টাকেই **Choose** করি?
     তাহলে এটাই যেহেতু সবথেকে বড় তাহলে এর ২গুন **(x+x= 2x) array** তে থাকার প্রশ্নই আসে না।
  3. **But, array** তে কিন্তু সব **negative elements** ও থাকতে পারে। অর্থাৎ, কোনো **positive** নাই।
     Ex - -2 -2 -1 -1 - এখানে যদি বড় **element** ২বার নি ও , **-1-1 = -2** মান কিন্তু বাড়ার বদলে কমবে, কারণ **- negative** মানের যোগফল মান কমায়।
     তাহলে, **2. no observation** টা আমাকে সবথেকে ছোট **element** এর জন্যেও চিন্তা করতে হবে, যেহেতু, **negative** মানের যোগফল মান কমায়, সেটাও **array** তে থাকার প্রশ্ন আসে না, **2 no.** এর বিপরীত।

- **Implementation:**
  Array থেকে **min , max element** বের করে নিব, তারপর **- observation -** এর শর্তগুলো **check** করব। **array -** তে আছে কিনা জানতে **set use** করতে পারি।

- **Code:**

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

#define ll long long

int main(){
    int tc=1;
    cin >> tc;
    while(tc--){
        int n;
        cin >> n;
        vector<ll> a(n);
        set<ll> st;
        ll mn = LLONG_MAX, mx = -LLONG_MAX;
        for(int i=0; i<n; i++){
            cin >> a[i];
            st.insert(a[i]);
            mn = min(mn,a[i]);
            mx = max(mx,a[i]);
        }

        if(st.find(2*mn) == st.end()){
            cout << mn << " " << mn << endl;
            continue;
        }

        if(st.find(2*mx) == st.end()){
            cout << mx << " " << mx << endl;
            continue;
        }

        cout << -1 << endl;
    }
    return 0;
}
/*   Author: Hridoy Barua (CS Instructor Phitron)   */
```

Problem G - Make same

- **Question Analysis:**

আমাকে ৩ টা **string** দেওয়া আছে **n length** এর । আমার কাজ হলো, ৩ টাকেই **identical** করা, অর্থাৎ **-** কোনো **string -** এ , একসাথে **- 1 & 0** থাকতে পারবে না।
পুরো **string -** এ হয়, **1** থাকবে , নাহলে **0** থাকবে।

Operationঃ

যেকোনো ২ টা **string** নাও, এর যেকোনো ২টা **index i,j (0 ≤ i, j <N)** এর **value swap** কর।

যদি, **identical** করা সম্ভব না হয়, **-1 output** করতে হবে। করা গেলে, **minimum operation** বলতে হবে।

- **Observation:**

1. ৩ টা **string** মিলে **total 1 and total 0 -** কে অবশ্যই **n** দিয়ে **divisible** হতে হবে।
   নাহলে **direct -1**।

2. এখানে কি কি **scenario** হতে পারে?

000	**111**	**already** এমন থাকলে,
000	**111**	**ans = 0**
000	**111**	

000	**000**	এরকম বানাতে হবে।
000	**111**	
111	**111**	

এমন **scenario** হলে, আমরা হয়, শুধু একটা **string -** এ **1** অথবা শুধু একটা **string -** এ **0** , এদেরকে **Target** করতে পারি।

3. কারণ, যেহেতু, সব **1** অথবা **0** , শুধু ১-টা **string-** এই আছে, যদি সব **1** অথবা **0**, অই একটা **string -** এ নিয়ে আসা মানে, **0** অন্য **string 2** টাতে **auto** বসে যাওয়া।
4. এখন কোন **string** এ সব **1** অথবা **0**, এমন করব?অবশ্যই **-**
   যদি **1** করতে চাই, যে **string** এ সবথেকে কম **0** সেই **string**।
   যদি **0** করতে চাই, যে **string** এ সবথেকে কম **1** সেই **string**।

এতে **operation** কম লাগবে।

- **Implementation:**

৩ টা **string** থেকেই **0 , 1 count** করে নিব।
Observation -1 আগেই **handle** করে নিব, যাতে **-1** এর প্যারা খাইতে না হয় আর।
তারপর **observation - 4** করব।

- **Code:**

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

#define ll long long

int main(){
    int tc=1;
    cin >> tc;
    while(tc--){
        int n;
        cin >> n;
        string a,b,c;
        cin >> a >> b >> c;

        int a0 = count(a.begin(),a.end(),'0'), a1 = n-a0;
        int b0 = count(b.begin(),b.end(),'0'), b1 = n-b0;
        int c0 = count(c.begin(),c.end(),'0'), c1 = n-c0;

        if((a0+b0+c0)%n!=0 && (a1+b1+c1)%n!=0){
            cout << -1 << endl; return;
        }
        if((a0+b0+c0)==n){
            cout << min({(n-a0),(n-b0),(n-c0)}) << endl;
        }
        else if((a1+b1+c1)==n){
            cout << min({(n-a1),(n-b1),(n-c1)}) << endl;
        }
        else{
            cout << 0 << endl;
        }
    }
    return 0;
}
/*   Author: Hridoy Barua (CS Instructor Phitron)   */
```