## سوال A: 1472A

- برای حل این سوال، ماکسمیم باری را که میتوان یه کاغذ را به کاغذهای کوچیکتر تقسیم کرد به دست میاوریم. اگر n کمتر و یا مساوی این عدد بود، اینکار ممکن است و در غیر اینصورت، اینکار ممکن نیست.

C++:

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

int main(){
    ll t;
    cin >> t;
    while(t--){
      ll w, h, n, res = 1;
      cin>>w>>h>>n;
      ll pro = w * h;
      while(pro % 2 == 0){
          pro /=  2;
          res *= 2;
      }
      if( n <= res)
          cout<<"YES"<<endl;
      else cout<<"NO"<<endl;
    }
}
```

Python:

```
for i in[*open(0)][1:]:
    w,h,n=map(int,i.split())
    w*=h
    q=1
    while w%2==0:q*=2;w//=2
    print("YNEOS"[n>q::2])
```

## سوال B: 894A

- ❖ راه اول شمردن همه‌ی حالات ممکن با استفاده از سه حقله‌ی تو در تو است:

C++:

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    cin>>s;
    int ans = 0;
    int n = s.size();
    for (int i = 0; i < n; i++) {
       if(s[i] == 'Q') {
           for (int j = i+1; j <n ; j ++) {
               if(s[j] == 'A') {
                   for (int k = j+1; k < n; k++) {
                       if(s[k] == 'Q')
                               ans++;

                   }
               }
           }
       }
    }
    cout<<ans;
    return 0;
}
```

❖ راه دوم پیدا کردن کاراکترهای A در جمله و استفاده از تعداد Qهای قبل و بعد از ان برای به دست اوردن تعداد حالات کلی است.

C++:

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;
    cin>>s;
    int ans = 0;
    int n = s.size();
    int pre[n] , post[n];
    for (int i = 0; i < n ;i++)
        post[i] = pre[i] = 0;

    if(s[0] == 'Q')
        pre[0] = 1;
    else pre[0] = 0;
    for (int i = 1; i < n; i++) {
        if(s[i] == 'Q')
            pre[i] = pre[i-1]+1;
        else pre[i] = pre[i-1];
    }
    if(s[n-1] == 'Q')
        post[n-1] = 1;
    else
        post[n - 1] = 0;

    for (int i = n - 2; i >= 0; i--) {
        if(s[i] == 'Q')
            post[i] = post[i+1]+1;
        else post[i] = post[i+1];
    }
    for (int i = 1 ; i < n-1; i++) {
        if(s[i] == 'A'){
            ans += (pre[i] * post[i]);
        }
    }
    cout<<ans;
}
```

Python

```
k=input()
print(sum(k[:i].count("Q")*k[i:].count("Q") for i in range(len(k)) if k[i]=="A"))
```

## سوال C: 1213B

- ما اعداد زوج را با حرکات آزاد و بدون هزینه میتوانیم بین مختصات های زوج و اعداد فرد را میتوانیم بدون هزینه بین مختصات های زوج جا به جا کنیم، بنابراین بررسی میکنیم که تعداد مختصات های زوج بیشتر است یا فرد و هرکدام بیشتر بود، مقصد نهایی را از ان نوع تعیین میکنیم و سپس تمام مختصات هایی که نوعی غیر از ان دارند، باید یک حرکت اضافه انجام دهند تا به مقصد برسند، بنابراین تعداد کل حرکات لازم منیمیم تعداد مختصات های زوج و فرد است.

C++:

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
    int n;
    cin>>n;
    long long a[n];
    int e = 0, o = 0;
    for (int i = 0 ;i < n; i++) {
        cin>>a[i];
        if(a[i] % 2 == 0) e++;
        else o++;
    }
    cout<<min(e,o);
}
```

Python:

```python
input()
a=[0,0]
for x in input().split():a[int(x)%2]+=1
print(min(a))
```

## سوال D: 1348B

- با توجه به اینکه لازم نیست تعداد insert ها را مینیمم کنیم، میتوانیم آرایه ی a را یک آرایه با اعداد متفاوتی که در آرایه ورودی وجود داره تصور کنیم. اگر تعداد این اعداد کمتر از k است، تا زمانی که به تعداد k برسد یک عدد ثابت مانند صفر را یک را به آن اضافه میکنیم. حال میتوانیم این آرابه را n بار پشت سر هم قرار دهیم. اگر تعداد اعداد متفادت در آرابه ورودی بیشتر از k است جواب نداریم و باید -1 چاپ کنیم .

C++:

```
#include <bits/stdc++.h>
using namespace std;
void solve(){
  int N,K;
  cin>>N>>K;
  set<int>s;
  for (int i=0;i<N;i++){
    int a;
    cin>>a;
    s.insert(a);
  }
  //if more than K distinct numbers, print -1
  if (s.size()>K){
    cout<<-1<<endl;
    return;
  }
  cout<<N*K<<endl;
  for (int i=0;i<N;i++){
    //print the distinct numbers
    for (int b:s)
      cout<<b<<' ';
    //print the extra 1s
    for (int j=0;j<K-(int)s.size();j++)
      cout<<1<<' ';
  }
  cout<<endl;
}

int main(){
  int t; cin>>t;
  while (t--)
    solve();
}
```

## سوال E: 124B

- به تعداد n و k در این مسئله دقت کنید. با توجه به اینکه ماکسیسمم آنها 8 است، میتوانیم تمام !8 حالت برای اعداد را حساب کنیم و مینیمم اختلاف ماکسیمم و مینیمم آنها را به دست بیاوریم.

C++:

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

int main(){
  ll n,k,ans = LONG_LONG_MAX;
  string num[8];
  ll p[] = {0,1,2,3,4,5,6,7};
  cin >> n >> k;
  for(ll i=0;i<n;i++)cin >> num[i];
  do{
    ll mx = -10;
    ll mn = LONG_LONG_MAX;
    for(ll i=0;i<n;i++){
      ll nu = 0;
      for(ll j=0;j<k;j++){
        nu = nu*10+(num[i][p[j]]-'0');
      }
      mx = max(mx,nu);
      mn = min(mn,nu);
    }
    ans = min(ans,mx-mn);
  }while(next_permutation(p,p+k));
  cout<<ans<<endl;
  return 0;
}
```