سوال A: 1294A

- اینکه در نهایت همه افراد به اندازه مساوی سکه داشته باشند، به این معناست که جمع سکه‌های فعلی یعنی A,B,C و سکه های جدید یعنی n باید بر ۳ بخش پذیر باشد.
- اما این تنها شرطی که باید چک شود نیست و برای اینکه در نهایت همه باید دقیقاً یک سوم این مجموع را داشته باشند، در ابتدا باید کم‌تر یا مساوی این مجموع سکه را داشته باشند.

C++:

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

int main(){
    ll  t;
    cin >> t;
    while(t--){
        ll a,b,c,n;
        cin >> a >> b >> c >> n;
        ll sum = a+b+c+n;
        cout<<(sum%3==0 && a<=sum/3 && b<=sum/3 && c<=sum/3? "YES":"NO")<<endl;
    }
    return 0;
}
```

Python:

```
for _ in range(int(input())):
    a,b,c,n = map( int,input().split())
    s = a+b+c+n
    print("YES" if s%3==0 and a<=s//3 and b<=s//3 and c<=s//3 else "NO")
```

سوال B: 1295A

- اول از همه ، نیازی به استفاده از هیچ رقمی غیر از 1 و 7 نداریم. اگر از رقم دیگری استفاده می کنیم ، از 4 یا بیشتر بخش تشکیل شده است ، بنابراین می توان آن را با 2 , 1 جایگزین کرد و عدد بیشتر می شود. به همین دلیل نیازی به استفاده بیشتر از یک 7 هم نداریم: اگر دو تا داریم ، می توانیم آنها را با۳ تا ۱ جایگزین کنیم و عدد بزرگ‌تر میشود.
- پس جواب دنباله ای از یک ها، یا یک ۷ و سپس دنباله ای از یک خواهد بود،از ۷ فقط وقتی استفاده میکنیم که n عددی فرد باشد.

C++:

```
#include <bits/stdc++.h>
using namespace std;
long long n,t;
int main(){
    cin>>t;
    for(int i=0;i<t;i++){
        cin>>n;
        if(n%2==1){
            cout<<7;
            for(int j=0;j<n-3;j+=2){
                cout<<1;
            }
        }
        else{
            for(int j=0;j<n;j+=2){
                cout<<1;
            }
        }
        cout<<endl;
    }
}
```

Python

```
for _ in range(int(input())):
    n = int(input())
    print('1'*(n//2) if n%2==0 else '7'+'1'*((n-3)//2))
```

سوال C: 75A

- با یک سؤال پیاده‌سازی طرفیم، بنابرین دقیقاً کاری که در متن سؤال گفته شده را انجام میدهیم و میبینیم که آیا معادله همچنان برقرار میماند یا خیر و سپس با توجه با برقرار ماند ن یا نماندن، جواب مناسب چاپ میکنیم.

C++:

```
#include <bits/stdc++.h>
using namespace std;
const int maxN = 100 + 321;
long long make_zero(long long n)
{
  long long t = 0;
  while(n > 0)
    {
      if(n %10)
    {
      t *= 10;
      t += n % 10;
    }
      n/=10;
    }
  while(t > 0)
    {
      n *= 10;
      n += t % 10;
      t /= 10;
    }
  return n;
}
int main()
{
  long long a, b;
  cin >> a >> b;
  long long sum = a + b;
  a = make_zero(a);
  b = make_zero(b);
  sum = make_zero(sum);
  if(sum == a + b)
    cout << "YES" << endl;
  else
    cout << "NO" << endl;
  return 0;
}
```

Python:

```
a = input()
b = input()
print("YES" if int(str(int(a)+int(b)).replace('0','')) == int(a.replace('0',''))+in
t(b.replace('0','')) else "NO")
```

سوال D: 1324B

- توجه به تعداد ورودی همیشه یه کار مفیده، این تعداد میتونه به ما ایده هایی برای حل سؤال بده،
- مثلاً اگه ببینیم تعدادطوری هست که با توجه به زمان، سؤال به راه حلی از مرتبه زمانی log(n) نیاز داره، میتونیم حدس بزنیم که باید طوری سؤال رو حل کنیم که هر دفعه بخش بزرگی از جواب‌ها ممکن حذف شن،یعنی مثلاً از یه چیزی شبیه باینری سرچ استفاده کنیم، یا اگه دیدیم اندازه ورودی طوری هست که حتی راه حل با مرتبه زمانی n^3 یا مرتبه ها بزرگتر هم کافیه، پس شاید یکی از راه‌های سؤال اینه که همه حالت‌های ممکن رو چک کنیم و ببینیم که آیا جواب بینشون هست یا نه.
- توی این سؤال هم ما یه تعداد تست کیس داریم، t=100 و یه n که توی هر کدوم از تست های متفاوته،اما یه تضمینی داده شده،اینکه مجموع n در کل تست کیس ها بیشتر از ۵۰۰۰ نشه.
- پس میتونیم تقریباً اینطوری فکر کنیم که که کلاً یه تست کیس ۵۰۰۰ تایی داریم.
- با توجه به اینکه عدد بزرگی نیست،پس شاید بتونیم بیایم همه حالت‌های ممکن رو چک کنیم و ببینیم چیزی که سؤال خواسته رو در آرایه داریم یا نه.چنین کاری چقدر زمان میگیره؟
- نیاز داریم که برای هر عدد توی آرایه،از دو تا عدد بعد شروع کنیم و ببینیم عددی مثلش پیدا میکنیم یا نه، اگه پیدا کردیم حله و جواب بله است و اگر نه جواب خیر است.
- یعنی دو تا حلقه تودرتو داریم، و حتی اگر هرکدوم n بار هم اجرا بشن، 25000000=5000*5000 در محدویت زمانی که سؤال داره به مشکل نمیخوره و این راه حل که از O(n^2) هست هم برای این سؤال جواب میده.پس خیلی دنبال راه حل سخت نمیگردیم و خیلی راحت میریم و همه حالت‌ها رو چک میکنیم.

Python:

```python
for _ in range(int(input())):
n=int(input())
a=[int(x) for x in input().split()]
f=0
for i in range(len(a)):
    if a[i] in a[i+2:]:
    f=1
    break
print("YES" if f else "NO")
```

C++:

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

int main(){
    ll t;
    cin >> t;
    while(t--){
        ll n;
        cin >> n;
        std::vector<ll> num(n);
        for(ll i=0;i<n;i++)cin >> num[i];
        bool ok = false;
        for(ll i=0;i<n-2;i++){
            for(ll j=i+2;j<n;j++){
                if(num[i]==num[j]){
                    ok=true;
                    break;
                }
            }
            if(ok)break;
        }
        cout<<(ok ? "YES":"NO")<<endl;
    }
    return 0;
}
```

سوال E: 343B

- اگر جایی دو قطعه سیم هم جریان کنار هم باشند،میتوانیم این قطعه را تکان دهیم و پس از این حرکت هم اگر باعث شویم دو قطعه هم جریان کنار هم قرار گیرند، دوباره میتوانیم این بخش جدید را هم تکان دهیم.
- برای حل این سؤال از یک stack استفاده میکنیم(خشت یک رو هم ان شا الله خوندید 😊)
- و برای هر کارکتر رشته ورودی اگر استک خالی بود یا کارکتر با کارکتری که روی استک است یکسان نبود،کارکتر جدید را در استک پوش میکنیم و اگر یکسان بود، کارکتر روی استک را پاپ میکنیم و چیزی را پوش نمیکنیم.
- در نهایت اگر استک خالی شده بود،جواب مثبت و در غیر این صورت منفی است.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main(){
    stack<ll> pm;
    string s;
    cin >> s;
    for(ll i=0;i<s.length();i++){
        if(pm.empty()||pm.top()!=s[i]){
            pm.push(s[i]);
        }
        else{
            pm.pop();
        }

    }
    pm.empty()? cout<<"Yes"<<endl:cout<<"No"<<endl;
    return 0;
}
```

Python:

```python
W,st = list(input()),[]
for i in W:
    if len(st) and st[-1]==i:
        st.pop()
    else:
        st.append(i)
print('No') if len(st) else print('Yes')
```