سوال A: 92A

https://codeforces.com/problemset/problem/92/A

عبارت sum%m به این معناست که ممکن است m یعنی تعداد چیپس ها انقدر زیاد باشد که بتوان چند دور زد. بنابراین با sum%m باقی مانده را بدست میاوریم که همان تعداد چیپس در دور آخر است.

حالا با یک حلقه for حساب میکنیم که چه موقع و با چه تعداد چیپس نمی توانیم ادامه دهیم.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n, m;
    cin>>n>>m;
    int sum=n*(n+1)/2;
    m = m%sum;
    for(int i=1; i<=n; i++){
        if(m>=i){
            m -= i;
        }
        else{
            cout<<m;
            return 0;
        }
    }
    cout<<m;
}
```

Python:

```python
n, m = map(int, input().split())
m %= (n+1)*n//2
for i in range(1, n+1):
   if m >= i: m -= i
print(m)
```

سوال B: 1559B

https://codeforces.com/problemset/problem/1559/B

اگر یک مربع در کنار مربع "؟" رنگ شده است، مربع "؟" را با رنگی مخالف مربع مجاور رنگ شده، رنگ میکنیم. در این صورت اثبات میشود که حداقل مقدار نامطلوب بدست می آید.

C++:

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

int main(){
   int t;
   cin>>t;
   while(t--){
       int n, x = -1;
       cin>>n;
       string s;
       cin>>s;
       for(int i = 0; i < n; i++) if(s[i] != '?') {x = i; break;}
       for(int i = x-1; i >= 0; i--) s[i] = (s[i+1] == 'R'?'B':'R');
       for(int i = x+1; i < n; i++) if(s[i] == '?') s[i] = (s[i-1] == 'R'?'B':'R');
       cout<<s<<endl;
    }
}
```

Python:

```python
for _ in range(int(input())) :
    n=int(input())
    s=input()
    while '?' in s :
        if s.count("?") == n:
            s = "BR"*n
            s = s[:n]
        s=s.replace('?B','RB')
        s=s.replace('?R','BR')
        s=s.replace('B?','BR')
        s=s.replace('R?','RB')
    print(s)
```

سوال C: 139A

https://codeforces.com/problemset/problem/139/A

اگر تعداد کل صفحات از تعداد صفحات روز دوشنبه بیشتر نباشد، پاسخ دوشنبه است. در غیر این صورت می توانیم عدد دوشنبه را از کل کم کنیم و به روز سه شنبه برویم. اگر سه شنبه کافی نیست کم می کنیم و تا چهارشنبه ادامه می دهیم و به همین ترتیب. ما مطمئن هستیم که بیش از N هفته نخواهد گذشت، زیرا هر هفته حداقل یک صفحه خوانده می شود. پیچیدگی O(N)

C++:

```cpp
#include <iostream>
using namespace std;
int p, mod = 1, d[8];
int main (){
    cin >> p;

    for (int i = 1; i <= 7; i++)
        cin >> d[i];

    while (p > 0){
        p -= d[mod];
        if (p <= 0)
            break;
        else if (mod == 7)
            mod = 1;
        else
            mod++;
    }
    cout << mod;
}
```

Python:

```python
n=int(input())
l=list(map(int,input().split()))
i=0
while n>l[i]:
    n-=l[i]
    i=(i+1)%7
print(i+1)
```

سوال D: 706B

https://codeforces.com/problemset/problem/706/B

کافی است آرایه مربوط به قیمت یک بطری برای هر مغازه را سورت کنیم. سپس با باینری سرچ روی این آرایه، ایندکس iای را بدست آوریم که x[i]<=m باشد و i بزرگترین ایندکس ممکن باشد.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
const int num = 100000+5;
int x[num];
int main(){
    int n, q; cin>>n;
    for(int i=0; i<n; i++){
        cin>>x[i];
    }
    cin>>q;
    sort(x, x+n);
    while(q--){
        int m; cin>>m;
        int indx = upper_bound(x, x+n, m) - x;
        cout<<indx<<endl;
    }
}
```

Python:

```python
import bisect;
input()
x=sorted(map(int ,input().split()))
for i in' '*int(input()):
   print(bisect.bisect(x,int(input())))
```

سوال E: 1490E

https://codeforces.com/contest/1490/problem/E

چگونه می توان یک بازیکن را بررسی کرد که آیا می تواند قهرمان شود؟ بدیهی است که او باید در تمام بازی ها شرکت کند (در غیر این صورت تعداد توکن های حریفان را افزایش خواهیم داد). بنابراین می توانید همه افراد را مرتب کنید و با ضعیف ترین ها حریصانه بازی کنید. چنین بررسی پس از مرتب سازی در زمان خطی کار می کند، بنابراین ما یک راه حل برای $O(n^2)$ دریافت کردیم.

ساده ترین راه حل برای این مسئله ، جستجوی دودویی برای پاسخ است. ما همه بازیکنان را بر اساس تعداد توکن هایی که دارند مرتب می کنیم. بیایید ثابت کنیم که اگر بازیکن i بتواند برنده شود، بازیکن i+1 نیز می تواند برنده شود (اعداد پس از مرتب سازی داده می شوند). اگر بازیکن i توانست برنده شود، بر اساس استراتژی بالا، او می‌توانست همه بازیکنان روی پیشوند [0...i+1] را شکست دهد. بازیکن i+1 همچنین می تواند همه این بازیکنان را شکست دهد زیرا حداقل به همان اندازه توکن دارد. حالا هر دو بازیکن باید همه حریفان را با اعداد [i+2...n] شکست دهند و تعداد تراشه‌هایی که هر دو بازیکن دارند برابر با مجموع اولین اعداد i+1 در آرایه است. بنابراین اگر بازیکن i یک استراتژی داشته باشد، بازیکن i+1 می تواند از همان استراتژی استفاده کند.

از این رو پاسخ مشکل این است - suffix مرتب شده آرایه ورودی. می توانید این suffix را با استفاده از جستجوی باینری و بررسی زمان خطی پیدا کنید.

*این مشکل یک راه حل کاملا خطی (پس از مرتب سازی) نیز دارد.

C++:

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

int main(){
    ll t;
    cin >> t;
    while(t--){
        ll n,ans=0,sum=0,min=0;
        cin >> n;
        std::vector<ll> num(n);
        std::vector<ll> nums(n);
        for(ll i=0;i<n;i++){
            cin >> num[i];
            nums[i]=num[i];
        }
        sort(nums.begin(),nums.end());
        min=nums[0];
        for(ll i=0;i<n-1;i++){
            sum+=nums[i];
            if(sum<nums[i+1])min=nums[i+1];

        }
        for(ll i=0;i<n;i++){
            if(num[i]>=min)ans++;
        }
```

```
        cout<<ans<<endl;
        for(ll i=0;i<n;i++){
            if(num[i]>=min){
                cout<<i+1<<' ';
            }
        }
        cout<<endl;
    }
}
```

Python:

```
def win(pos : int, a : list):
    power = a[pos]
    for i in range(len(a)):
        if i == pos:
            continue
        if power < a[i]:
            return False
        power += a[i]
    return True

t = int(input())
while t > 0:
    t -= 1
    n = int(input())
    a = list(map(int, input().split(' ')))
    b = a.copy()
    a.sort()
    l = -1
    r = n - 1
    while r - l > 1:
        m = (l + r) // 2
        if (win(m, a)):
            r = m
        else:
            l = m
    winIds = []
    for i in range(n):
        if b[i] >= a[r]:
            winIds.append(i + 1)
    print(len(winIds))
    print(*winIds)
```