سوال A: 1551A

- بیایید متغیرهای c1 و c2 را با مقدار یکسان [n3] مقدار دهی کنیم. سپس ما باید باقی مانده تقسیم n بر ۳ را جمع آوری کنیم. اگر باقی مانده برابر ۰ باشد ، نیازی به جمع آوری چیز دیگری نداریم زیرا متغیرهای c1 و c2 قبلاً روی پاسخ صحیح تنظیم شده اند: |c1 – c2| = 0 زیرا c1 = c2 و هیچ مقدار مطلق نمی‌تواند کمتر از ۰ باشد. در غیر این صورت |c1 – c2| != 0 زیرا c1==c2 و n = c1*2+c2 = 3*c1 که در صورت بخش پذیر نبودن n بر ۳ غیر ممکن است.

- اگر باقیمانده برابر ۱ است ، پس باید با استفاده از یک سکه ۱ مقدار ۱ را جمع آوری کنیم ، بنابراین c1 را یکی افزایش میدهیم. در این مورد ، c1 = c2+1 ، از این رو |c1 – c2| = 1 ، این مقدار نمی‌تواند کمتر از ۱ باشد ، همانطور که در بالا ثابت شد.

- اگر باقیمانده برابر ۲ است ، پس باید با استفاده از یک سکه ۲ عدد ۲ را جمع آوری کنیم ، بنابراین دهید c2 را به علاوه یک میکنیم. در این مورد ، c2 = c1+1 ، از این رو |c1 – c2| = 1.

C++:

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

int main(){
	ll t;
	cin >> t;
	while(t--){
		ll n;
		cin >> n;
		cout<<n/3+(n%3==1)<<' '<<n/3+(n%3==2)<<endl;
	}
	return 0;
}
```

Python:

```
n=int(input())
for i in range(n):
   a=int(input())
   print(a//3+(a%3==1),a//3+(a%3==2))
```

سوال B: 709A

- سوال پیاده سازی است و برای حل آن باید دقیقا کاری که گفته شده را انجام دهیم , پرتقال های با سایز مناسب را آب بگیریم و هر وقت تعداد از d بیشتر شد، یکی به جواب اضافه کنیم و دوباره اندازه بگیریم.

C++:

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

int main(){
	ll n,b,d;
	cin >> n >> b >> d;
	ll in=0;
	ll ans =0;
	for(ll i=0;i<n;i++){
		ll o;
		cin >> o;
		if(o<=b)in+=o;
		if(in>d){ans++;in=0;}
	}
	cout<<ans<<endl;
	return 0;
}
```

Python

```
N,B,D=map(int,input().split())
A=list(map(int,input().split()))
S=0
SUM=0
for I in range(N):
	if A[I]<=B:
		SUM+=A[I]
		if SUM>D:
			SUM=0
			S+=1
print(S)
```

سوال C: 322B

- اگر "دسته گل مخلوط" وجود نداشته باشد ، پاسخ r/3 + g/3 + b/3 خواهد بود. یک نکته مهم این است که: همیشه یک راه حل بهینه با کمتر از ۳ دسته گل مخلوط وجود دارد.

- چرا؟ : هنگامی که ۳ دسته گل مخلوط داشته باشیم ، می توانیم آن را به (۱ دسته گل قرمز + ۱ دسته سبز + ۱ دسته گل آبی) تغییر دهیم.

- بنابراین می توانیم تعداد ۰ ، ۱ ، ۲ دسته گل مخلوط داشته باشیم و بقیه ۳ دسته گل را تک رنگ بسازیم. بین سه حالت بالا آن حالتی که بیشترین نتیجه را داشت خروجی میدهیم.

C++:

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

int main(){
	ll r,g,b;
	cin >> r >> g >> b;
	ll ans = 0;
	for(ll i=0;i<=min(3LL,min(r,min(b,g)));i++){
		ans = max(ans,(r-i)/3+(b-i)/3+(g-i)/3+i);
	}
	cout<<ans<<endl;
	return 0;
}
```

Python:

```python
r,g,b = map(int,input().split())
print (max(i+(r-i)//3+(g-i)//3+(b-i)//3 for i in
range(min(r+1,g+1,b+1,3))))
```

سوال D: 230B

- نکته ای که وجود دارد این است که تنها اعدادی سه مقسوم علیه دارند که مجذور یک عدد اول باشند.(چرا؟)
- پس میتوانیم با استفاده از یک غربال اراتستن اعداد اول را تا مجذور ۱۲۸۱۰ حساب کنیم و بعد از آن تنها چک کنیم که آیا مجذور عدد ورودی داده شده اول هست یا نه.

(اگر با غربال اراتستن آشنایی ندارید، سرچ و کلاس رفع اشکال فراموش نشود)

CPP:

```
include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mx = 1e6+2;
int main(){
	std::vector<ll> num(mx,true);
	num[1] = false;
	for(ll i=2;i<mx;i++){
		if(num[i]){
			for(ll j=i+i;j<mx;j+=i){
				num[j] = false;
			}
		}
	}
	ll n;
	cin >> n;
	while(n--){
		ll t;
		cin >> t;
		if(sqrt(t)==int(sqrt(t))&&num[int(sqrt(t))])
			cout<<"YES"<<endl;
		else
			cout<<"NO"<<endl;
	}
	return 0;
}
```

Python:

```
n=1000000
a=[1]*n
s=set()
for i in range(2,n):
	if a[i]:
		s.add(i*i)
		for j in range(i*i,n,i):
			a[j]=0
input()
for x in map(int,input().split()):
	print(["NO","YES"][x in s])
```

سوال E: 568A

- میتوانیم با استفاده از یک غربال اراتستن اعداد اول را پیدا کنیم و برای دسترسی داشتن به تعداد اعداد اول تا عددی مثلn ، در خانه n ام تعداد اعداد اول تا خانه n-1 ام را ذخیره کنیم و اگر خود n اول بود،یکی به آن بیفزاییم.
- برای داشتن تعداد اعداد palindrome تا عددی مثل n هم میتوانیم از ایده ای شبیه ایده بالا استفاده کنیم.
- در نهایت هم با یک پیمایش تا حداکثر بازه اعداد و استفاده از مقادیر از پیش محاسبه شده، جواب سوال را پیدا میکنیم.(سعی کنید پیچیدگی زمانی کد را پیدا کنید.)

C++:

```cpp
#include <bits/stdc++.h>

using namespace std;
typedef long long ll;
typedef long double ld;
const ll mx = 3*1e6+10;
ll pr[mx],pa[mx],ans=-1;
bool is_prime[mx];

bool isp(ll num){
	string s = to_string(num);
	ll n = s.length();
	for(ll i=0;i<n/2;i++){
		if(s[i]!=s[n-1-i])return false;
	}
	return true;
}

int main(){
	for(ll i=2;i<mx;i++){
		if(!is_prime[mx]){
			for(ll j=i+i;j<mx;j+=i){
				is_prime[j]=true;
			}
		}
		pr[i] = pr[i-1]+(!is_prime[i]);
	}
	pa[0]=0;
	for(ll i=1;i<mx;i++){
		pa[i] = pa[i-1]+isp(i);
	}
	ll p,q;
	cin >> p >> q;
	for(ll i=1;i<mx;i++){
		if(ld(pr[i])<=((ld(p)/q)*pa[i])){
			ans = i;
		}

	}
	ans!=-1 ? cout<<ans<<endl:cout<<"Palindromic tree is better than splay tree"<<endl;
	return 0;
}
```

Python:

```python
N = int(2e6+3)
is_prime = [0] * N
def sieve():
   is_prime[1] = 1;
   for i in range(2, N):
       if is_prime[i] == 1:
           continue
       for j in range(2*i, N, i):
           is_prime[j] = 1

p, q = map(int, input().split())
rub = 0
pi = 0
sieve()
for i in range(1, N):
   if is_prime[i] == 0:
       pi += 1
   if str(i) == str(i)[::-1]:
       rub += 1
   if pi * q <= rub * p:
       res = i
print(res)
```