سوال A: 490A

❖ بیشترین تعداد تیم هایی که میتوان ساخت با کمترین تعداد ۱ یا ۲ یا ۳ ها برابر است. برای چاپ حالت های مختلف نیز، هر کدام از ۱ و ۲ و ۳ ها را در یک لیست ذخیره میکنیم و آنها را به ترتیب چاپ میکنیم.

C++:

```cpp
#include <bits/stdc++.h>
#include <iostream>
using namespace std;
int main(){
    int t,a;
    cin>>t;
    vector<int> v1,v2,v3;
    for(int i=0;i<t;i++){
          cin>>a;
          if(a==1) v1.push_back(i+1);
          else if(a==2) v2.push_back(i+1);
          else v3.push_back(i+1);
    }
    auto mini =min(v1.size(),v2.size());
    mini=min(mini,v3.size());
    cout<<mini<<endl;
    for(int i=0;i<mini;i++)
        cout<<v1[i]<<' '<<v2[i]<<' '<<v3[i]<<endl;
}
```

Python:

```python
n = int(input())
lst = list(map(int, input().split()))
l1, l2, l3 = list(), list(), list()
for i in range(n):
    if lst[i] == 1:
        l1.append(i+1)
    elif lst[i] == 2:
        l2.append(i+1)
    elif lst[i] == 3:
        l3.append(i+1)
m = min(lst.count(1), lst.count(2), lst.count(3))
print(m)
for i in range(m):
    print(l1[i], l2[i], l3[i])
```

سوال B: 1462C

- میدانین بزرگترین رقم، ۹ است و برای به دست آوردن کوچکترین رقم بهتر است تعداد ارقام آن کمترین باشد. درنتیجه اگر از بزرگترین رقم یعنی ۹، شروع به کم کردن از n کنیم (با شرط اینکه ۹ در پاسخ رقم آخر باشد) میتوانیم به کوچکترین رقم با مجموع n برسیم. پس تا زمانی که n>0 است هر بار از i=9 شروع میکنیم و i را از آن من میکنیم و مقدار i را کاهش میدهیم.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main(){
    ll t, n;
    cin >> t;
    while (t--){
        cin >> n;
        ll i = 9;
        vector<ll> num;
        if (n > 45)
            cout << "-1" << endl;
        else{
            if (n < 10)
                cout << n << endl;
            else{
                while (n > i && n > 0){
                    n -= i;
                    num.push_back(i);
                    i--;
                }
                if (n > 0)
                    num.push_back(n);
                sort(num.begin(), num.end());
                for (ll i = 0; i < num.size(); i++)
                    cout << num[i];
                cout << endl;
            }
        }
    }
    return 0;
}
```

Python:

```python
for _ in range(int(input())):
    m = ''
    x = int(input())
    if x > 45:
        print(-1)
    else:
        for i in range(9, 0, -1):
            if x - i <= 0:
                break
            m = str(i) + m
            x -= i
        if x > 0:
            m = str(x) + m
        print(m)
```

سوال C: 1324A

- از آنجایی که مقدار اضافه شده به یک ستون در هر گام 2 واحد و زوج است، برای اینکه بتوان همه ستون ها را 0 کرد باید اختلاف ارتفاع هر دو ستون مقداری زوج باشد و این یعنی ارتفاع تمام ستون ها یا زوج و یا فرد باشد.

C++:

```cpp
#include <bits/stdc++.h>
using  namespace std;
int main() {
    int t,n,x,y;
    bool flag;
    cin >> t;
    while(t--){
        flag = true;
        cin >> n;
        cin >> x;
        for (int i = 1 ; i < n ; ++i){
            cin >> y;
            if (y%2 != x%2)
                flag = false;
        }
        if (flag)
            cout << "YES" << endl;
        else
            cout << "NO" << endl;
    }
}
```

Python:

```python
for qq in range(int(input())):
    n = int(input())
    a = list(map(lambda x: int(x)%2, input().split()))
    print("YNEOS"[len(set(a))==2::2])
```

سوال D: 1182A

- با شکل داده شده فقط n های زوج را میتوان پر کرد. از طرفی هر ۲*۳ را میتوان به ۲ طریق مختلف پر کرد. باید تعداد ۲*۳ های مستطیل را پیدا کنیم و حالا مختلف را به دست بیاوریم.

C++:

```cpp
#include <bits/stdc++.h>
#include <iostream>
using namespace std;
int main()
{
    int n;
    cin>>n;
    if(n%2 == 1) cout<<0<<endl;
    else cout<<(int)pow(2, n/2);
}
```

Python:

```python
n = int(input())
if n % 2 == 1:
    print(0)
else:
    print(int(pow(2, n / 2)))
```

سوال E: 279B

- برای به دست آوردن بیشترین تعداد کتاب قابل مطالعه در زمان t، نیاز داریم محاسبه کنیم با شروع از هر کتاب، چه تعداد کتاب قابل مطالعه است. برای محاسبه‌ی این تعداد، ابتدا هر خانه از آرایه رو به صورت جمع خانه های قبلی، به علاوه‌ی مقدار این خانه در نظر میگیریم. سپس با شروع از کتاب اول، محاسبه میکنیم که چه تعداد کتاب قابل مطالعه است و سپس وقتی این تعداد به محدودیت زمانی رسید، با عوض کردن مقدار l، شروع به محاسبه ی تعداد کتاب‌هایی میکنیم که میتوان با شروع از کتاب دوم مطالعه کرد و به همین منوال ادامه میدهیم و در هربار بررسی، تعداد ماکسیمم کتاب قابل مطالعه را آپدیت میکنیم.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mx = 1e5+10;
ll b[mx],l;
int main(){
  ll n, t, ans=0;
  cin >> n >> t;
  for(ll i=1;i<=n;i++){
    ll in;
    cin >> in;
    b[i] = b[i-1] + in;
    if(b[i]-b[l]>t)
      l++;
    ans = max(ans,i-l);
  }
  cout << ans << endl;
  return 0;
}
```