به نام خالق یکتا

راه حل سوال های کانتست UICPC Round #19 (Div2)

سوال A: 318A

- عملیات را شبیه سازی میکنیم، به این صورت که دو متغیر به عنوان اشاره گر، یکی برای اشاره به اول آرایه و دیگری برای اشاره به آخر آرایه در نظر میگریم، و در هر مرحله بیشترین مقدار از بین مقادیری که این اشاره گر ها به آن‌ها اشاره میکنند را در نظر گرفته و به مجموع امتیازها کسی که نوبتش است اضافه میکنیم.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main(){
    ll n,s=0,d=0;
    cin >> n;
    std::vector<ll> num(n);
    for(ll i=0;i<n;i++)cin >> num[i];
    ll i=0,j=n-1;
    for(ll q=0;q<n;q++){
        if(num[i]<num[j]){
            q%2==0 ? s = s+num[j]:d = d+num[j];
            j--;
        }
        else{
            q%2==0 ? s = s+num[i]:d = d+num[i];
            i++;
        }
    }
    cout<<s<<' '<<d<<endl;
    return 0;
}
```

Python:

```python
n=int(input())
a=list(map(int,input().split()))
x=0
y=0
for i in range(n):
    t=max(a[0],a[-1])
    if (i%2!=0):
        x+=t
    else:
        y+=t
    a.remove(t)
print(y,x)
```

سوال B: 1472B

- اگر تعداد کل آبنبات‌ها فرد باشد، به این معنی که تعداد یک ها فرد باشد، نمیتوان انها را بین دو نفر تقسیم کرد. اگر به تعداد زوج 1 گرمی و به تعداد زوج 2 گرمی داشته باشیم، میتوانیم به راحتی ابنبات ها را به دسته های دوتایی تقسیم کنیم. اگر تعداد 1 گرمی ها زوج و تعداد 2 گرمی ها فرد باشد، برای اینکه به طور مساوی قابل تقسیم باشند باید حداقل 2 آبنات 1 گرمی داشته باشیم، زیرا 2 آبنبات یک گرمی را مانند یک، 2 گرمی در نظر میگیریم و این حالت تبدیل به حالتی میشود که گویی هر دو آبنبات به تعداد زوج موجود بودند.

C++:

```cpp
#include<bits/stdc++.h>
using namespace std;
int main(){
    int t,n,x;
    cin>>t;
    while(t--){
        cin>>n;
        int two=0,one=0;
        while(n--){ cin>>x; x==2 ? two++ : one++;}
        if(two%2==0 && one%2==0) cout<<"YES"<<endl;
        else{
            if(two %2!=0 && (one+2)%2==0 && one!=0) cout<<"YES"<<endl;
            else cout<<"NO"<<endl;
        }
    }
    return 0;
}
```

Python

```python
for _ in range(int(input())):
    n = int(input())
    l = list(map(int,input().split()))
    if(sum(l)%2==1):
        print("NO")
    else:
        print( "YES" if n%2==0 or l.count(1)>=2 else "NO")
```

سوال C: 1471A

❖ با توجه به

[a/x] + [b/x]:
1 - a//x + b//x + 0
2- a//x + b//x + 1
3- a//x + b//x + 2

a // x >>
خارج قسمت تقسیم
[] همان سقف

case 1:
[(a + b) / x] = [a/x] + [b/x]

case 2:
a = nx + A
b = Nx
>>
[(a + b) / x] = [(nx + A + Nx) / x] = [n + N + A /x] = a//x + b//x + 1

case 3:
a = nx + A
b = nx + B
>>
[(a + b) / x] = [(nx + A + Nx + B) / x] = [n + N + A /x + B / x]
>>
A /x < 1
B / x < 1
>> A/x + B/x
1- (A/x + B/x =< 1) >> [(a + b) / x] = a//x + b//x + 1
2 - (A/x + B/x > 1) >> [(a + b) / x] = a//x + b//x + 2

❖ میتوان ثابت کرد که:

$\lceil (a+b)/x \rceil \leq \lceil a/x \rceil + \lceil b/x \rceil$

❖ بنابرین کم ترین جواب برای وقتیست که ابتدا همه عناصر آرایه را جمع و سپس تقسیم را انجام دهیم وبیشترین حالت مربوط به حالتی است که اعداد را جداگانه تقسیم کنیم و سقف بگیریم و سپس جمع کنیم.

C++:

```cpp
#include<bits/stdc++.h>
using  namespace std;
long long n,t,mx,a,x,k,mn;
int main(){
    cin>>t;
    while(t--){
        cin>>n>>x;
        mx=mn=0;
        for(k=1;k<=n;k++){
            cin>>a;
            mx+=(a+x-1)/x;
            mn+=a;
        }
        cout<<(mn+x-1)/x<<" "<<mx<<endl;
    }
}
```

Python:

```python
import math as m
for _ in range(int(input())):
    n, x = map(int,input().split())
    l = list(map(int,input().split()))
```

```
    print(m.ceil(sum(l)/x),sum(m.ceil(e/x) for e in l))
```

سوال D: 1234B1

- در این سوال نیاز داریم که دقیقا عملیات شرح داده شده را انجام دهیم.
- استفاده از ساختمان های داده مناسب میتواند کار را بسیار راحت کند.
- برای شبیه‌سازی صف از یک Deque یا صف دو طرفه استفاده میکنیم و همانطور که در سوال گفته شده با اضافه شدن یک عنصر جدید، آن را به ابتدای صف اضافه میکنیم و اگر تعداد عناصر از k بیشتر شد، از آخر آن حذف میکنیم. برای اینکه بفهمیم آیا یک عنصر هم اکنون در صفحه در حال نمایش داده شدن است یا نه هم از یک ست اسفاده میکنیم که محتوای آن همواره عناصر حال حاضر صفحه باشند.

C++:

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

int main(){
    ll n,k;
    deque<ll> screen;
    cin >> n >> k;
    set<ll> is_in;
    for(ll i=0;i<n;i++){
        ll in;
        cin >> in;
        if(!is_in.count(in)){
            screen.push_front(in);
            is_in.insert(in);
        }
        if(screen.size()>k){
            is_in.erase(screen.back());
            screen.pop_back();
        }
    }
    cout<<screen.size()<<endl;
    while(!screen.empty()){
        cout<<screen.front()<<' ';
        screen.pop_front();
    }
    cout<<endl;
    return 0;
}
```

Python:

```
n, k = tuple(map(int, input().split()))
ids = list(map(int, input().split()))
S = []
for id in ids:
    if id not in S:
        S = [id] + S
    S = S[:k]
print(len(S))
print(" ".join([str(s) for s in S]))
```

سوال E: 797C

- در هر مرحله باید کوچک ترین حرف را در رشته های فعلی حفظ کنیم (این را می توان با نگه داشتن آرایه ای 26تایی از تعداد دفعات نمایش هر حرف در رشته و به روز رسانی آن در هر مرحله انجام داد). بیایید رشته t را stack بنامیم و از آن استفاده کنیم. حالا شما حروف را از s یکی یکی استخراج می کنید و در پشته قرار میدهید. تا زمانی که سر استک از همه حروف باقیمانده در s کوچکتر مساوی است آن را به پاسخ اضافه میکنیم. پس از خالی شدن s نیز هر چیزی که در استک باقیمانده به پاسخ اضافه میکنیم و چاپ میکنیم.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long int lli;
int main () {
    string s, t, ans;
    cin >> s;
    multiset<char> d;
    for (auto el : s) {
        d.insert(el);
    }
    int i = 0;
    while (i < (int)s.size()) {
        if (t.size() == 0) {
            t += s[i];
            i++;
        } else {
            auto it = d.lower_bound(t.back());
            if (it == d.begin()) {
                ans += t.back();
                t.pop_back();
                d.erase(it);
                if (t.size() > 0) {
                    d.insert(t.back());
                }
            } else {
                it = d.find(t.back());
                d.erase(it);
                t += s[i];
                i++;
            }
        }
    }
    while (t.size() > 0) {
        ans += t.back();
        t.pop_back();
    }
    cout << ans;
    return 0;}
```