به نام خالق یکتا

راه حل سوال های کانتست UICPC Round #20 (Div2)

سوال A: 1560A

- ابتدا همه اعدادی که طبق تعریف خوب هستند را درون یک وکتور می‌ریزیم.
- با توجه به اینکه k<=1000 است بزرگترین عدد خوب ۱۶۶۶ است و از لحاظ زمانی به مشکلی نمی‌خوریم.
- سپس برای هر تست کیس k را ورودی گرفته و عدد متناظر با آن را چاپ می‌کنیم.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll mx = 2001;
vector<ll> nums;
int main()
{
    for (ll i = 1; i < mx; i++)
    {
        if (i % 10 != 3 && i % 3 != 0)
            nums.push_back(i);
    }
    ll t;
    cin >> t;
    while (t--)
    {
        ll k;
        cin >> k;
        cout << nums[k - 1] << endl;
    }
    return 0;
}
```

Python:

```python
a=[i for i in range(9001) if i%3 and i%10!=3]
print(*[a[int(input())-1] for _ in range(int(input()))])
```

سوال B: 1555A

- توجه داشته باشید "سرعت" پخت ۱ تکه پیتزا برای همه اندازه ها یکسان است - ۱ تکه پیتزا به مدت ۲/۵ دقیقه.
- اگر n فرد باشد ، آن را یکی افزایش می دهیم (زیرا پیتزا فقط با تعداد زوج تکه پخته می شود). پس مقدار n همیشه زوج است. اگر n <6 ، پس برای چنین n ای پاسخ برابر با n = 6 است ، بنابراین می توانیم بگوییم n = max (n,6)، تا وقتی که n>=12 می توانیم یک پیتزای کوچک سفارش دهیم. در نهایت مقدار n برابر ۶ ، ۸ یا ۱۰ خواهد بود. این بدان معناست که برای هر n مجموعه ای از پیتزاها با دقیقاً n برش وجود خواهد داشت. پس پاسخ 2.5 * n است.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
  int t;
  cin >> t;
  while (t--) {
    long long n;
    cin >> n;
    cout << max(6LL, n + 1) / 2 * 5 << '\n';
  }
}
```

Python

```python
for _ in range(int(input())):
    print(max(6,int(input())+1)//2*5)
```

سوال C: 1195B

- در این سوال باید در واقع معادله $x(x+1)/2-(n-x)=k$ را حل کنیم و در نهایت n-x ر چاپ کنیم.
- میتوانین معادله را به روش ریاضی حل کنیم، اما یک راه حل پر کاربرد و زیبای دیگر استفاده از باینری سرچ است، به این صورت رو مقدار های ممکن جواب باینری سرچ بزنیم و با توجه به تغییرات مورد نیازش، بازه جواب را کوچکتر کنیم تا به جواب اصلی برسیم.

C++:

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

int main(){
    ll n,k;
    cin >> n >> k;
    ll l=-1,r=n+1;
    while(l<r-1){
        ll m = (l+r)/2;
        if((n-m)*(n-m+1)/2-m>k)l = m;
        else
            r= m;
    }
    cout<<r<<endl;
    return 0;
}
```

Python:

```python
import math
n, k = map(int, input().split())
q = int(-3 + math.sqrt(9 + 8*(n+k)))//2
r = n-q
print(r)
```

سوال D: 220B

- اول باید این رو متوجه بشیم که اگه a[i] بزرگتر از سایز ارایه باشه ممکن نیست که به تعداد خودش تکرار بشه و میتونیم این a[i] رو در نظر نگیریم.
- وقتی که این کار رو انجام بدیم عدد هایی که ممکنه داخل آرایه باعث تغییر جواب بشن تو رنج ۱ و اندازه ارایه هستن
- حالا کافیه که روی همه عددها پیمایش کنیم و اگه عددی تعداد تکرارش از مقدار خودش بیشتر بود متوجه میشیم که این عدد روی جواب تاثیر داره.
- ارایه p رو میسازیم که نشون میده عدد مورد نظر چند بار قبل از خونه i ام p تکرار شده.
- حالا روی همه کوئری ها for میزنیم و اون هایی که این عدد به جوابشون اضافه میکنه رو مشخص میکنیم و بهشون اضافه میکنیم.
- این کار رو برای همه عدد ها تکرار میکنیم تا جواب بدست بیاد.
- در مورد پیچیدگی زمانیش در بدترین حالت باید برای کوچیک ترین عدد ها یک بار روی کل ارایه پیمایش کنیم.
- پس اگه اعداد و تعداد تکرارشون به این صورت باشن ۱, ۲, ۳, ۴, ۵, ... , x پیچیدگی زمانیش به این صورته:

$1 + 2 + 3 + 4 + 5 + 6$,, $x = n$

$x * (x + 1) / 2 = n$
$x^2 + x = 2n$
$x^2 < 2n$
$x < sqrt(2n)$

$=> O(n * sqrt(2n)) = O(n * sqrt(n))$

C++:

```cpp
#include<iostream>
using namespace std;
const int N=101000;
int l[N],r[N],p[N],a[N],cnt[N],out[N];
int main()
{
    int n,q;
    cin>>n>>q;
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
        if(a[i]<N) cnt[a[i]]++;
    }
    for(int i=0;i<q;i++) cin>>l[i]>>r[i];
    for(int num=1;num<N;num++)
    {
        if(cnt[num]>=num)
        {
            p[0]=0;
            for(int i=1;i<=n;i++) p[i]=p[i-1]+(num==a[i]);
            for(int i=0;i<q;i++) out[i]+=((p[r[i]]-p[l[i]-1])==num);
        }
    }
    for(int i=0;i<q;i++) cout<<out[i]<<"\n";
    return 0;
}
```