## سوال A: 1353A

❖ اگر صرفا یک جایگاه داشته باشیم _n == 1_چون اطرافی ندارد جواب صفر است، اگر دو جایگاه داشته باشیم، بهترین حالت این است که یکی از جایگاه ها صفر باشد و دیگری برابر m باشد. برای سه جایگاه و بیشتر از آن بهترین حالت این است که یکی از جایگاه ها مقدار m را داشته باشد و دو جایگاه اطرافش مقدار صفر را.

C++:

```
#include <bits/stdc++.h>
using  namespace std;
typedef unsigned long long ull;
int main() {
    int t;
    cin>> t;
    while(t--){
        ull n, m;
        cin>>n>>m;
        if(n == 1)
            cout<<0<<endl;
        else if (n == 2) cout<<m<<endl;
        else  cout<< m * 2<<endl;

    }

}
```

Python:

```
for _ in range(int(input())):
    n, m = map(int, input().split())
    print(min(2, n-1)*m)
```

## سوال B: 1370A

- اگر بزرگترین مقسوم علیه را g در نظر بگیریم و با توجه به اینکه حداقل یکی از دو عدد از g بزرگتر است، میتوان گفت 2 * g کوچکترین مقداری است که میتوان با به دست آورد و بنابر این پاسخ n / 2 است.

C++:

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ull;
int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        int n;
        cin >> n;
        if (n % 2 == 0)
            cout << n / 2 << endl;
        else
            cout << (n - 1) / 2 << endl;
    }
}
```

Python

```
for s in[*open(0)][1:]:print(int(s)//2)
```

## سوال C: 805A

❖ از بین هر دوعدد متوالی، حتما یکی از انها به 2 بخش پذیر است، بنابراین داریم:

C++:

```
#include <bits/stdc++.h>
using namespace std;
typedef unsigned long long ull;
int main()
{
    ull l, r;
    cin >> l >> r;
    if (l == r)
        cout << l;
    else
        cout << 2;
}
```

Python:

```
l,r=input().split();print(2 if l!=r else l)
```

## سوال D: 1395B

- راه حل های مختلف و خلاقانه ای برای حل این سوال وجود دارد، مانند راه حل هایی که در کانتست با آنها سوال حل شد (راه حل شما چه بود؟ اشتراک گذاری کنید 🙂 )
- یکی از راه حل های ساده، حرکت دادن sx و sy به یکی از ضلع ها، و حرکت عمودی یا افقی است. به عنوام مثال، میتوانیم ابتدا به خانه (sy ,1) برویم، سپس به خانه (1 ,1) برویم و تا خانه (m ,1) حرکت کنیم و به خانه هایی در این ردیف که هنوز دیده نشده اند برویم. سپس یک سطر به پایین بیایم و اینکار را برای سطر بعدی تکرار کنیم. توجه داشته باشید که اگر روی سطر های زوج از راست به چپ حرکت میکنید، روی سطر های فرد باید از چپ به راست حرکت کنید.

C++:

```
#include <bits/stdc++.h>
using namespace std;
main(){
    int n, m, sx, sy;
    vector<pair<int, int>> res;
    bool grd[103][103];
    memset(grd, 0, sizeof(grd));

    cin >> n >> m >> sx >> sy;
    sx--;
    sy--;

    res.push_back({sx + 1, sy + 1});
    grd[sx][sy] = 1;
    res.push_back({1, sy + 1});
    grd[0][sy] = 1;

    for (int i = 0; i < n; i++){
        if (i % 2 == 0)
        {
            for (int j = 0; j < m; j++)
            {
                if (!grd[i][j])
                {
                    res.push_back({i + 1, j + 1});
                    grd[i][j] = 1;
                }
            }
        }
        else
        {
            for (int j = m - 1; j >= 0; j--)
            {
                if (!grd[i][j])
                {
                    res.push_back({i + 1, j + 1});
                    grd[i][j] = 1;
                }
            }
        }
    }

    for (int i = 0; i < res.size(); i++)
    {
        cout << res[i].first << " " << res[i].second << endl;
    }
```

## سوال E: 1549C

- برای حل این سوال، ممکن است با چالش های زیادی روبرو بشین. مثل نحوه ورودی گرفتن و ذخیره گراف. محاسبه خانه های vulnarable و.. . برای حل این سوال باید هر بار موقع ورودی گرفتن، ببینیم آیا این نوبل vulnarable است یا خیر.
- سوال به این صورت حل میشود:
- به تعداد m یال ها (friendship) ها دریافت میشود. و هر بار گرفتن:
- نوبل کوچکتر را پیدا میکنیم. اگر این نوبل قبلا vulnarable نبوده است، آن را vulnarable میکنیم و به تعداد vulnarable ها یکی اضافه میکنیم.
- به تعداد q دستورات را میگیریم
- برای دستور وارد کردن: دوباره؛ نوبل کوچکتر را پیدا میکنیم. اگر این نوبل قبلا vulnarable نبوده است، آن را vulnarable میکنیم و به تعداد vulnarable ها یکی اضافه میکنیم.
- برای دستور حذف کردن: ابتدا رابطه ی بین دو نوبل را حذف میکنیم. سپس هر دو نوبلی که میخواهیم رابطه friendship بین آنها را حذف کنیم بررسی میکنیم. هر کدام اگر vulnarable بود، همه ی نوبل های دیگر که با آن در ارتباط هستند را چک میکنیم، اگر بین آنها نوبلی پیدا شد که مقدار آن، بزگتر از نوبلی است که میخوایم حذف کنیم، آنگاه نوبا همچنان vulnarable است در غیر اینصورت دیگر vulnarable نیست و باید از تعداد vulnarable ها یکی کم کنیم.
- برای دستور process کردن: تنها لازم است تعداد کل نوبل ها را از vulnarable ها کم کنیم.

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
ll const mx = 2 * 1e5 + 10;
ll cnt[mx], n, m, ans = 0, q;
int main()
{
    cin >> n >> m;
    for (ll i = 0; i < m; i++)
    {
        ll u, v;
        cin >> u >> v;
        cnt[min(u, v)]++;
    }
    for (ll i = 1; i <= n; i++)
    {
        if (!cnt[i])
            ans++;
    }
    cin >> q;
    while (q--)
    {
        ll c;
        cin >> c;
        if (c == 1)
        {
            ll u, v;
            cin >> u >> v;
            cnt[min(u, v)]++;
            if (cnt[min(u, v)] == 1)
                ans--;
        }
        else if (c == 2)
        {
            ll u, v;
            cin >> u >> v;
            cnt[min(u, v)]--;
            if (!cnt[min(u, v)])
                ans++;
        }
        else if (c == 3)
        {
            cout << ans << endl;
        }
    }
    return 0;
}
```