سوال A: 1436B

- در ابتدا باید در نظر داشته باشیم که اعداد 0 و 1 **اول نیستند**.
- می دانیم که $1 + 1 = 2$ **عددی اول** است .
- پس در هر سطر از دو عدد 1 استفاده می کنیم و هر بار آن ها را به راست شیفت می دهیم .

C++:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
void rot(std::vector<ll> &num)
{
    ll t = num[0];
    for (ll i = 0; i < num.size() - 1; i++)
        num[i] = num[i + 1];
    num[num.size() - 1] = t;
}
int main()
{
    ll t;
    cin >> t;
    while (t--)
    {
        ll n;
        cin >> n;
        std::vector<ll> num(n, 0);
        num[n - 1] = 1;
        num[n - 2] = 1;
        for (ll i = 0; i < n; i++)
        {
            for (ll j = 0; j < n; j++)
            {
                cout << num[j];
                if (j < n - 1)
                    cout << ' ';
            }
            cout << endl;
            rot(num);
        }
    }
    return 0;
}
```

Python:

```python
for _ in range(int(input())):
    n = int(input())
    s = ["1", "1"] + ["0"] * (n - 2)
    for _ in range(n):
        print(" ".join(s))
        s = s[-1:] + s[:-1]
```

سوال B: 1097A

- کافیست کارت هایی که در دستمان داریم را با کارت روی میز مقایسه کنیم ، اگر **کارکتر مشترکی** با کارت روی میز وجود داشت "YES" را چاپ کنیم. در غیر اینصورت "NO"

C++:

```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
    string s, t;
    int i;
    cin >> s;
    for (i = 0; i < 5; i++)
    {
        cin >> t;
        if (s[0] == t[0] || s[1] == t[1])
        {
            cout << "YES";
            return 0;
        }
    }
    cout << "NO";
    return 0;
}
```

Python:

```python
card_table = input()
card_hand = input().split()
flag = False

for card in card_hand:
    if card[0] == card_table[0] or card[1] == card_table[1]:
        print("YES")
        flag = True
        break

if not flag:
    print("NO")
```

سوال C: 119A

❖ جهت حل سوال ، کافیست بازی گفته شده را **پیاده سازی** کنید :)

C++:

```cpp
#include <iostream>
using namespace std;

int gcd(int x, int y)
{
    return (x == 0) ? y : gcd(y % x, x);
}

int main()
{
    int a, b, n;
    cin >> a >> b >> n;
    int k = 0;
    while (n >= 0)
    {
        ++k;
        n -= gcd((k & 1) ? a : b, n);
    }
    if (k & 1)
        cout << 1;
    else
        cout << 0;
}
```

Python:

```python
import math

a, b, n = [int(i) for i in input().split()]
one = True

while n:
    if one:
        n -= math.gcd(a, n)
    else:
        n -= math.gcd(b, n)
    one = not one

if one:
    print(1)
else:
    print(0)
```

سوال D: 702A

- شمارنده ای را در نظر میگیریم و مقدار اولیه 1 را به آن میدهیم. اعداد ورودی را از اول پیمایش میکنیم؛ اگر عددمان از عدد قبلی خود بزرگتر بود ، یک واحد به شمارنده اضافه میکنیم و اگر چنین نبود شمارنده رو مجددا 1 (پس از چک کردن شرط ماکسیمم بودن ...) قرار داده و به همین ترتیب ادامه می دهیم.
- **ماکسیمم مقدار شمارنده مان** همان جواب مسئله است.

C++:

```cpp
#include <iostream>
#include <climits>
using namespace std;

int main()
{
    int n, res = 1, inp[2], mres = INT_MIN;

    cin >> n >> inp[0];
    for (int i = 0; i < n - 1; i++)
    {
        cin >> inp[1];

        if (inp[1] > inp[0])
            res++;

        else
        {
            mres = max(mres, res);
            res = 1;
        }

        inp[0] = inp[1];
    }

    mres = max(mres, res);
    cout << mres << endl;
    return 0;
```

Python:

```python
a, b, m, n = 0, 0, 0, input()
for i in map(int, input().split()):
    m = m+1 if i > b else 1
    a = max(a, m)
    b = i
print(a)
```

سوال E: 337A

- ابتدا عدد های سطر دوم ورودی - تعداد تکه های هر پازل - را به صورت **صعودی** مرتب می کنیم.
- بعد فرض میکنیم پازلی با کوچک ترین تعداد تکه از f[k] قسمت تشکیل شده است. روشن است که معلم n پازل را با تعداد تکه f[k] یا بیشتر از آن با حداقل فاصله های میان تکه ها انتخاب میکند که f[k] , f[k+1] , ... , f[k+n-1] می باشد.
- در این زیرمجموعه n تایی که انتخاب کردیم تفاوت بین کمترین و بیشترین تعداد تکه ها f[k+n-1]-f[k] است.
- پس برای انتخاب بهترین f[k] میتوانیم **هر k بین 1 تا m-n را امتحان کرده و کمترین فاصله** را پیدا کنیم.

C++:

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int main()
{
    int n, m, arr[50], ans;

    cin >> n >> m;

    for (int i = 0; i < m; i++)
        cin >> arr[i];

    sort(arr, arr + m);
    ans = arr[m - 1] - arr[0];

    for (int i = 0; i <= m - n; i++)
        ans = min(ans, arr[i + n - 1] - arr[i]);

    cout << ans << endl;
    return 0;
}
```

Python:

```python
n, m = map(int, input().split())
A = sorted(list(map(int, input().split())))
MIN = A[n - 1] - A[0]
for i in range(m - n + 1):
    MIN = min(A[i + n - 1] - A[i], MIN)
print(MIN)
```

سوال F: 1514B

- می دانیم که بازه اعداد ما $0 \leq \text{elements} \leq 2^k - 1$ است.
- از این موضوع می توان نتیجه گرفت که عدد ما **k بیت** دارد جهت به دست آوردن مقدار بیشینه ، تمامی این بیت ها باید 1 باشند .
- حال با توجه به شرط سوال جهت صفر شدن & اعداد باید **دقیقا یکی از بیت ها 0** شود (چرا ؟) .
- **با توجه به این نکات** تعداد حالت های ممکن ما برابر n^k است .

C++:

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

void solve()
{
    int n, k;
    cin >> n >> k;
    int m = 1e9 + 7;
    int an = 1;
    for (int i = 0; i < k; ++i)
    {
        an *= n;
        an %= m;
    }
    cout << an << "\n";
}

signed main()
{
    ios_base::sync_with_stdio(false);
    int t;

    cin >> t;
    while (t--)
        solve();

    return 0;
}
```

Python:

```python
mo = 1000000000 + 7
for __ in range(int(input())):
    n, k = map(int, input().split())
    print((n**k) % mo)
```