#pragma GCC optimze("Ofast")

#pragma GCC target("avx,avx2,fma")

#include<bits/stdc++.h>

#define ll long long int

#define fri(s,e) for(int i=s;i<e;i++)

#define frj(s,e) for(int j=s;j<e;j++)

#define pb push\_back

#define mp make\_pair

#define pf push\_front

#define veci vector<int>

#define vecb vector<bool>

#define f first

#define sec second

#define db double

const int mod = 1e7;

#define fast ios\_base::sync\_with\_stdio(false);cin.tie(NULL);cout.tie(NULL);

using namespace std;

#define nl “\n”

Vector<int> array\_of\_vector[100001];//makes an array of vectors

int c[10][10]={{0}};//format for setting all zero

// to avoid modulus of negative number

For operations where subtraction is required, it is better to write-

(a-b+mod)%mod  
then- (a-b)%mod  
To avoid modulus of negative values

/\*

//dfs

vector<int> lst[100001];

int visited[100001]={0};

bool compare(vector<int> &a,vector<int> &b)//without & it will give tle for llarge numbers

{

return a[1]<b[1];

}

void dfs(int node)

{

visited[node]=1;

for(int child:lst[node])

{

if(!visited[child])

dfs(child);

}

}

//printing the subsequences using backtracking

void printSubSeqRec(string str, int n,

           int index = -1, string curr = "")

{

    // base case

    if (index == n)

        return;

    cout << curr << "\n";

    for (int i = index + 1; i < n; i++) {

        curr += str[i];

        printSubSeqRec(str, n, i, curr);

        // backtracking

        curr = curr.erase(curr.size() - 1);

    }

    return;

}

\*/

//position of 1stt set bit is given by 1+log2(n&-n)and fisrt set bit is given by n&-n;

//counts set bits in O(1)

private static int meth(int a)  
{  
a^=a>>16;  
a^=a>>8;  
a^=a>>4;  
a&=0xf;  
return (0x6996 >> a) & 1;  
}//returns 1 for odd and 0 for even

//unsigned short int is of 16 bit

//important formula a+b=( a Xor B) + 2\*(a AND b)  
ll power(ll x,ll y)  
{  
    ll res=1;  
    while(y>0)  
    {  
        if(y%2==1)  
            res=(res\*x)%mod;  
        x=(x\*x)%mod;  
        y=y/2;  
    }  
    return res;  
}

void SieveOfEratosthenes(int n)

{

    bool prime[n+1];

    memset(prime, true, sizeof(prime));

    for (int p=2; p\*p<=n; p++)

    {

        if (prime[p] == true)

        {

            // Update all multiples of p

            for (int i=p\*2; i<=n; i += p)

                prime[i] = false;

        }

    }

  }

int main()

{

int t;cin>>t;

while(t--)

{

}

return 0;

}