Dhaka University of Engineering & Technology, Gazipur.
Team Name: Still Alive

Math formula:

i. Sn = (n/2) * (2*a+(n-1)*d)
ii. Last term l = a+(n-1)*d
iii. Arithmetic progression a+b = 2*b
iv. **$\Sigma n^2 = [n(n+1)(2n+1)]/6$**
v. Even: **$\Sigma(2n)^2 =[2n(n+1)(2n+1)]/3$**
vi. **Odd: $\Sigma(2n-1)^2 = [n(2n+1)(2n-1)]/3$**
vii. Stars and Bars: (m-1)C(n-1). if **0 allowed then (n+m-1)C(m-1)**

Number Theory:

GCD algorithm using loop:

```
while(b){
   r = a%b;
   a = b;
   b = r;
}
gcd = a;
```

gcd(a,b) = gcd(a-b,b)
gcd(a,0) = a

Sieve of Eratosthenes:

```
#define M 1000000
bitset <M> primes;
void sieve(){
   primes[0]=1;primes[1]=1;
   for(int i=2;i*i<=M;i++){
      if(primes[i]==0){
         for(int j=i*i;j<=M;j+=i){
            primes[j]=1;
         }
```

Prime Factorization:

```
vector<int> factors;
void primeFactorize(int n){
   sieve(n);//Generate prime 2 to
sqrt(n+7)
   for(int i = 0; i<prime.size(); i++){
      if(n%prime[i]==0){
         while(n%prime[i]==0){
            n/=prime[i];
            factors.pb(prime[i]);
         }
      }
   }
    if(n>1) factors.pb(n);
}
```

Count the no. of divisors upto n

```
int count_divisor(int n){
   if(n==1||n==2) return n;
   int divisors = 1;
   if(n%2==0){
      int cnt = 1;
      while(n%2==0){
         n/=2;
         cnt++;
      }
      divisors*=cnt;
   }
   if(n>1){
      for(int i=3;i*i<=n;i+=2){
         if(n%i==0){
            int cnt = 1;
            while(n%i==0){
               n/=i;
               cnt++;
            }
            divisors*=cnt;
         }
      }
   }
   if(n>1){
      divisors+=divisors;
   } return divisors;}
```

Dhaka University of Engineering & Technology, Gazipur.
Team Name: Still Alive

List of Prime divisors:

```
vector<int> div;
primes[0]=1;primes[1]=1;
for(int i=2;i*i<=n;i++){
if(primes[i]==0){
for(int j=i*3;j<=n;j+=i){
      div[j].pb(i);
      primes[j]=1;
    }
  }
}
```

List of all divisors of all numbers:

```
vector<int> div[n+1];
for(int i=1;i<=n;i++){
   for(int j=i;j<=n;j+=i){
      div[j].pb(i);
   }
}
```

Q: Find common divisors of two numbers.
Ans: First find gcd of those two numbers then find the number of divisors of gcd. The numbers can be more than 2. Then also find the ans using same logic.

Q: Given two number a and b, find the number of numbers from L to R that are divisible by both a and b.
Ans: if(L==1) R/lcm(a,b)
       else (R/lcm(a,b) - (L-1)/lcm(a,b)

Q: Check if a number is fibonacci.
Ans: if(5*n*n+4)==Perfect square or (5*n*n-4)==Perfect square) then n is a member of fibonacci sequence.

Bit Manipulation:

Q: Print xor of all elements 1 to n.
Ans: n%4==0 | n, n%4==1 | 1, n%4==2 | n+1, n%4==3 | 0.

Q: Print value of xor (L to R)
Ans: Xor(1 to R) ^ (1 to L-1)

Inclusion and Exclusion:
Q: How many numbers are there in range 1-500 which are divisible by either 2 or 3 or 5.
Ans: (500/2)+(500/3)+(500/5)-(500/2*3)-(500/3*5)-(500/2*5)+(500/2*3*5) ->Odd numbers divisor is additive and even number divisor is subtractive.

Set the ith bit.

```
mask = 1ll<<i;
ans = n|mask;
```

Clear the ith bit.

```
mask = 1ll<<i;
n = n&(~mask);
```

Flip the ith bit.

```
mask = 1ll<<i;
n = n^mask;
```

Count the number of set bit.

```
while(n){
    cnt++;
    n = n&(n-1);
}
```

Power set algorithm.

```
for(i=0;i<(1ll<<n);i++){
   for(j=0;j<n;j++){
      if(i&(1<<j)) arr[j];
   }
   cout << "\n";
}
```

Two Pointer.00010001001=max=4->How many removable before 1 including 1.

```
for(i=0,j=0;j<n;j++){
    if(s[j]=='1'){
        ct=max(ct,j-i);
        i=j;
```

Dhaka University of Engineering & Technology, Gazipur.
Team Name: Still Alive

Q: “*….****..*” count the maximum “.” in a row.**

```
 for(i=0;i<n;i++){
       if(s[i]=='.'&&i!=n){
          cnt++;
       }else{
          ans=max(ans,cnt);
          cnt=0;
       }
    }
```

Lower bound in binary search.

```
ll l=0,r=v.size()-1;
ll ans=-1,mid;
   while(l<=r){
      mid=l+(r-l)/2;
      if(v[mid]>=target){
         ans=mid;
```

NcR using Fermat little theorem.

```
ll power(ll x,ll y,ll p){
   ll res = 1; // Initialize result
   x = x % p;
   while (y > 0){
      if (y & 1) res = (res * x) % p;
      y = y >> 1;
      x = (x * x) % p;
   }
   return res;
}
// Returns n^(-1) mod p
ll modInverse(ll n,ll p){
   return power(n, p - 2, p);
}

// Returns nCr % p using Fermat's little
theorem.
ll nCrModPFermat(ll n,ll r,ll p){
   if (n < r) return 0;
   if (r == 0) return 1;
   ll fac[n + 1];
   fac[0] = 1;
   for (int i = 1; i <= n; i++) fac[i] =
(fac[i - 1] * i) % p;
   return (fac[n] * modInverse(fac[r],
p) % p
         * modInverse(fac[n - r], p) % p)
      % p;
}
```

KMP:

```
// Add some code
#include<bits/stdc++.h>
using namespace std;
int main(){
   int t; cin>>t;
   for(int tt =1; tt<=t; tt++){
      string s,p;
      cin>>s>>p;
      int cnt=0;
      //Constructing the pi table.
      vector<int> pi(p.size());
      int now = -1;
      pi[0] = -1;
      for(int i = 1; i<p.size();i++){
         while(now!=-1 &&
p[now+1]!=p[i]){
            now = pi[now];
         }
         if(p[now+1]==p[i]) now++;
         pi[i] = now;
      }
      // Now KMP
      now = -1;
      for(int i = 0; i<s.size(); i++){
         while(now!=-1 &&
p[now+1]!=s[i]){
            now = pi[now];
         }
         if(p[now+1]==s[i]) now++;
         if(now==p.size()-1) cnt++;
      }
      cout << "Case " << tt << ": " <<
cnt <<"\n";
   }
   return 0;
```

Cumulative Hashing.

```
#include<bits/stdc++.h>
#define ll long long int
#define Mod 1000000007
#define Mod2 998244353
#define limit 1000008
using namespace std;
void cumforwardhashing(string s,ll
base,ll mod,ll A[])
{
  ll i,n=s.size();
  A[0] = s[0]-'a'+1;
for(i=1; i<n; i++)
```

```
   {
      A[i] = ((A[i-1]*base)+s[i]-
'a'+1)%mod;
   }
}
void cumbackwordhashing(string s,ll
base,ll mod,ll A[])
{
   ll i,n=s.size();
   A[n-1] = s[n-1]-'a'+1;
   for(i=n-2; i>=0; i--)
   {
      A[i] = ((A[i+1]*base)+s[i]-
'a'+1)%mod;
   }
}
int  main()
{
   int i;
   string s;
   cin >> s;
   ll A[s.size()+5],B[s.size()+5];
   cumforwardhashing(s,29,Mod,A);
   cumbackwordhashing(s,29,Mod,B);
   for(i=0; i<s.size(); i++)
      cout <<A[i]<<" ";
   cout <<endl;
   for(i=0; i<s.size(); i++)
      cout <<B[i]<<" ";
   cout <<endl;
   return 0;
```

Knapsack:

```
ll n,w;
v64 wt,v;
ll dp[101][100001];
void solve(){
   ll weight,value,i,j,ans = -
LONG_LONG_MAX;
    cin>>n>>w;
    forn(i,n){
        cin>>weight>>value;
        wt.pb(weight);
        v.pb(value);
    }
    dp[0][0] = 0;
    dp[0][wt[0]] = v[0];
    for(i = 1; i<n; i++){
        dp[i][0] = 0;
        for(j = 1; j<=w; j++){
            //Not Pick
            dp[i][j] = dp[i-1][j];
            //Pick if j>=wt[i]
            if(j>=wt[i]) dp[i][j] =
max(dp[i][j],v[i]+dp[i-1][j-wt[i]]);
        }
    }
    for(j = 0; j<=w; j++){
        ans = max(ans, dp[n-1][j]);
    }
    cout << ans << "\n";
}
```

Diajkstra:

```
vector<pair<ll,ll>> adj[N];
void solve(){
    ll n,m,temp,ans=0,i,j,k,a,b,c;
    cin>>n>>m;
    forn(i,m){
          ll u,v,w; cin>>u>>v>>w;
          adj[u].pb({v,w});
          adj[v].pb({u,w});
    }

    ll dis[n+1],prev[n+1];

forn(i,n+1) dis[i] = inf;
          ll src = 1;
          dis[src] = 0;
          priority_queue<pair<ll,ll>> pq;
          pq.push({0,src});
          while(!pq.empty()){
                    auto x = pq.top();
                    pq.pop();
                    ll u = x.se;
                    ll d = -x.fi;
                    if(dis[u]<d) continue;
                    for(auto y: adj[u]){
                              ll v = y.fi;
                              ll w = y.se;
                              if(dis[v]>d+w){
                                        dis[v] =
d+w;

          pq.push({-dis[v],v});
                                        prev[v] =
u;//for printing the shortest path
```

```
                }
            }
        }

        if(dis[n]==inf) cout <<
"IMPOSSIBLE\n";
        else{
            ll destination = n;
            v64 vv;
            while(destination!=src){

        vv.pb(destination);
                destination =
prev[destination];
            }
            vv.pb(src);
            reverse(all(vv));
            cout << vv.size() << nn;
            forn(i,vv.size()) cout <<
vv[i] << " ";
            cout << nn;
        }

}
```

BFS Shortest Path Printing:

```
vector<ll> adj[N];
int vis[N];


void solve(){
   ll n,m,temp,ans=0,i,j,k,a,b,c;
   cin>>n>>m;
   ll prev[n+1];
   forn(i,m){
        ll u,v; cin>>u>>v;
            adj[u].pb(v);
            adj[v].pb(u);
   }
   queue<ll> q;
   ll src = 1;
   q.push(src);
   vis[src] = 1;
   while(!q.empty()){
        ll u = q.front();
        q.pop();
        for(auto v: adj[u]){
            if(!vis[v]){
                vis[v] = 1;
                prev[v] = u;
                q.push(v);
            }
        }
   }

   ll destination = n;
   v64 path;
   while(destination!=src){
        path.pb(destination);
        destination = prev[destination];
   }
   path.pb(src);
   reverse(all(path));
   cout << path.size() << nn;
   forn(i,path.size()){
        cout << path[i] << " ";
   }
   cout << nn;

}
```

Path Finding of a tree.

Find Cycles in a directed graph:

```
//Verifying if a topological sort is
possible or not.
//vis=0(Not visited)   vis=1(Exploring)
vis=2(Finished)

vector<ll> adj[N],vis(N);
ll nd = -1;//Indicates Cycle.
stack<ll> st;
void dfs(ll u){
   if(vis[u]==1){
      nd = u;
      return;
   }
   st.push(u);
   vis[u] = 1;
   for(auto v : adj[u]) {
      if(vis[v]!=2) dfs(v);
      if(nd!=-1) return;
   }
   vis[u] = 2;
   st.pop();
}
```

Cycle Detection and Printing Undirected:

```
v64 adj[N];
v64 vis(N);
ll nd=-1;//Node which completes the
cycle.
stack<ll> st;//Stack for storing the
visited nodes.
```

```
void dfs(ll u, ll prev){
   if(vis[u]){
      nd = u;//If a node is already
visited and it is not the prev visited
node then cycle formed.
      return;
   }
   st.push(u);
   vis[u] = 1;
   for(auto v: adj[u]){
      if(v!=prev){
         dfs(v,u);
         if(nd!=-1) return;//Return here
because we don't want to pop stack
data if a cycle has already been found

      }
   }
   st.pop();//Pop the stack when
backtracking.
}
```

Tree Diameter:

```
v64 adj[N];

v64 dis(N); //Stores depth of tree for
each node from the root.

void dfs(ll u, ll p, ll d){
   dis[u] = d;
   for(ll v: adj[u]){
      if(v!=p){
         dfs(v,u,d+1);
      }
   }
}
```

Union Find or DSU:

```
ll parent[N],sz[N];
priority_queue<ll> pq;
void make_set(ll n){
   for(ll i = 0; i<=n; i++){
      parent[i] = i;
      sz[i] = 1;
   }
}

ll find_set(ll u){
   if(parent[u]==u) return u;
   return parent[u] =
find_set(parent[u]);
}

bool union_set(ll u, ll v){
   ll a = find_set(u);
   ll b = find_set(v);

   if(a!=b){//If      a== b then cycle
detected.
      if(sz[a]>sz[b]){
         swap(a,b);
      }
      parent[a] = b;
      sz[b]+= sz[a];
      pq.push(sz[b]);
      return true;
   }else return false;
}

void solve(){
   ll n,m,temp,ans=0,i,j,k,a,b,c,u,v;
   cin>>n>>m;
   ans = n;
   make_set(n);
   forn(i,m){
      cin>>u>>v;
      if(union_set(u,v)) ans--;
      cout<<ans << " " << pq.top() <<
nn;
   }
}
```

Graph Bicoloring:

```
//Finding odd length cycle using graph
bicoloring aka trying to color the graph
using only two
//color such that no adjacent node
have the same color.

v64 adj[N];
ll color[N];
ll flag = 1;
void dfs(ll u, ll c){
   if(color[u]){
      if(color[u]!=c){
         flag = 0;//Odd length cycle
found.
         return;
      }
      return;
   }
   color[u] = c;
   for(ll v: adj[u]){
      dfs(v,3-c);// either color 1 or color
2.
```

```
    }
}

void solve(){
    ll n,m,temp,ans=0,i,j,k,a,b,c,u,v;
    cin>>n>>m;
    forn(i,m){
        cin>>u>>v;
        adj[u].pb(v);
        adj[v].pb(u);
    }
    for(i =1; i<=n; i++){
        if(!color[i]){
            dfs(i,1);
        }
    }
    if(!flag) cout << "IMPOSSIBLE";
    else for(i = 1; i<=n; i++) cout <<
color[i] << " ";
    cout << nn;

}
```

Trailing zeros of Facatorial.

```
    int count = 0;

    // Keep dividing n by
powers of
    // 5 and update count
    for (int i = 5; n / i >=
1; i *= 5)
        count += n / i;

    return count;
```

Coin Combination I:

```
ll n,x,dp[N];

v64 coin;

ll f(ll amount){
    if(amount==x) return 1;
    if(amount>x) return 0;

    if(dp[amount]) return dp[amount];
    ll res = 0;
    for(int i = 0; i<n; i++){
        res =
(res+f(amount+coin[i]))%mod;
    }
    return dp[amount] = res;
}

void solve(){
    ll temp;
    cin>>n>>x;
    forn(i,n){
        cin>>temp;
        coin.pb(temp);
    }
    cout << f(0) << nn;
}
```

Coin Combination II:

```
ll n,x;
v64 coin;
ll dp[N][101];

ll f(ll amount, ll indx){
        if(indx==n){
                if(amount==x) return 1;
                else return 0;
        }
        if(amount>x) return 0;
        if(dp[amount][indx]) return
dp[amount][indx];
        ll res = 0;
        res = (res +
f(amount+coin[indx],indx))%mod;
        res = (res+
f(amount,indx+1))%mod;

        return dp[amount][indx]=res;
}

void solve(){
    ll m,temp,ans=0,i,j,k,a,b,c;
    cin>>n>>x;
    forn(i,n){
        cin>>temp;
        coin.pb(temp);
    }
    cout << f(0,0) << nn;
}
```

Knapsaack 1:

```
ll n,w;
vector<pair<ll,ll>> v;
vector<v64> dp(1005,v64 (105,-1));

ll f(ll cur, ll indx){
        if(indx==n){
                if(cur<=w) return 0;
        }
    if(cur>w) return -inf;

        if(dp[cur][indx]!=-1) return dp[cur][indx];

        ll ret = max(v[indx].se+f(cur+v[indx].fi,indx+1),f(cur,indx+1));
        return dp[cur][indx]= ret;

}


void solve(){
    ll m,temp,ans=0,i,j,k,a,b,c;

    cin>>n>>w;
    forn(i,n){
        cin>>a>>b;
        v.pb({a,b});
    }
    cout << f(0,0) << nn;
    }
```

PBDS:

```
Ordered_set os;
Os.insert(1);
Cout << *os.find_by_order(1);//Print index 1
Cout << os.order_of_key(4); //Return the index of 4 or count of smaller number of elements
```

Template:

```
#include <bits/stdc++.h>
using namespace std;

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

typedef long long ll;
typedef long double ld;
typedef pair<ll,ll> p64;
typedef vector<ll> v64;
typedef vector<p64> vp64;
ll mod = 1e9+7;
const ll N = 500005;
double eps = 1e-12;
ll dr[] = {1,-1,0,0};
ll dc[] = {0,0,1,-1};
#define forn(i,e) for(ll i = 0; i < e; i++)
#define rforn(i,s) for(ll i = s; i >= 0; i--)
#define pb push_back
#define fi first
#define se second
#define nn "\n"
#define inf 2e18
#define setbit(x,k) (x|= (1LL<<k))
#define checkbit(x,k) ((x>>k)&1)
#define clearbit(x,k) (x&= ~(1LL<<k))
#define yes cout << "YES\n"
#define no cout << "NO\n"
#define fast_cin() ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL)
#define all(x) (x).begin(), (x).end()
#define ordered_set tree<ll, null_type,less<ll>, rb_tree_tag,tree_order_statistics_node_update>

void solve(){
    ll n,m,temp,ans=0,i,j,k,a,b,c,u,v;
}
int main()
{
    fast_cin();
    ll t = 1;
    cin >> t;
    for(int it=1;it<=t;it++) {
        solve();
    }
    return 0;
}
```