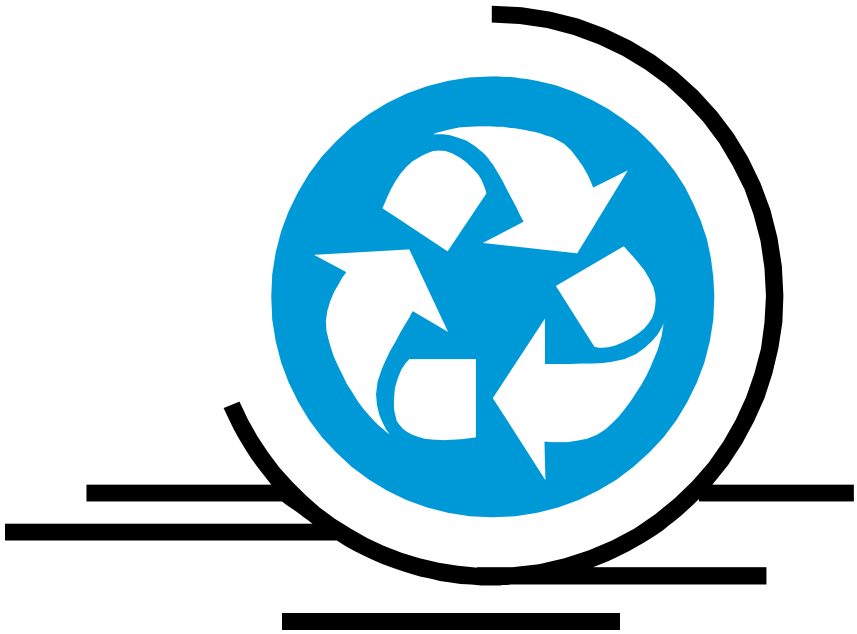

Document prepared by **Forman khan**

PROGRAMMING

Notes

Template

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

// loop and test case
#define ft    int t; scanf("%d", &t); for(int c=1; c<=t; c++)

//memset function
#define setz(a)  memset(a, 0, sizeof(a))
#define setneg(a)  memset(a, -1, sizeof(a))

#define sf  scanf
#define pf  printf

// integer input
#define isc(a)   scanf("%d", &a)
#define isc2(a,b)  scanf("%d%d", &a,&b)
#define isc3(a,b,c)  scanf("%d%d%d", &a,&b,&c)

// long and unsinged long long integer input
#define lsc(a)   scanf("%ld", &a)
```

```
#define lsc2(a,b)  scanf("%ld%ld", &a,&b)
#define llsc(a)  scanf("%lld", &a)
#define llsc2(a,b)  scanf("%lld%lld", &a, &b)
#define llsc3(a,b,c)  scanf("%lld%lld%lld", &a, &b, &c)
#define ullsc(a)  scanf("%llu", &a)
#define ullsc2(a,b)  scanf("%llu%llu", &a, &b)

// double input && output
#define  dsc(a)   scanf("%lf", &a)
#define  dpf2(a)   printf("%.2f", a)
#define  dpf3(a)   printf("%.3f", a)

//print function
#define  ipf(a)   printf("%d", a)
#define  lpf(a)  printf("%ld", a)
#define  llpf(a)  printf("%lld", a)
#define ullpf(a) printf("%llu", a)
#define nl printf("\n")

//Case printing
#define  casepf(a)  printf("Case %d: ", a)
#define  casestr(a, b)  printf("Case %d: %s\n", a,b)
```

```
//floor && ceil
#define  fl(a)  floor(a)
#define  cl(a)   ceil(a)


#define  pb(a)  push_back(a)
#define  mp(a,b)  make_pair(a,b)
typedef   pair<int, int> pii;
typedef  long  long  int  ll;
typedef  unsigned  long  long  ull;

//GCD
template <class T>
inline T gcd(T x,T y)
{
   T mod;
   while(x%y)
   {
      mod = x%y;
      x = y;
      y = mod;
   }
   return y;
}
```

```
int kdx[] = {2,2,-2,-2,1,-1,1,-1};
int kdy[] = {1,-1,1,-1,2,2,-2,-2};

int edx[] = {1,-1,0,0,1,1,-1,-1};
int edy[] = {0,0,1,-1,1,-1,1,-1};

int fdx[] = {1,-1,0,0};
int fdy[] = {0,0,1,-1};

int main()
{
}
```

Input Output Techniques

There are several functions that are used to take input and print output efficiently. Some
of them are discussed here.

scanf:

It is the simplest of all. Most of the time, it is used to take inputs. Due to the problems of taking string as input, sometime gets() function is used instead of it.

Example 1:

The important part of ***scanf*** *function is its return value.*

a=scanf("%d %d",&b,&c);

a. The result will be 2 if two integer variables are given correctly as input.

b. The result will be 1 if first input is given correctly and second one is not given or given illegal input(such as string) or first one contains some illegal input. For example:

56 abc

53ab 34

45 ^Z

c. The result will be 0 if first input is given incorrectly.

d. The result will be -1 if end of file (EOF) is found without getting any input.

Example 2:

Suppose the problem states,

"There will be two integers in every line of the input and terminated by EOF (end of file)."

Then you should write

```
int a,b;
while(scanf("%d %d",&a,&b)==2) //two inputs are taken correctly
{
// rest of the work
}
```

Example 3:

If the problem states,

"There will be three numbers (not necessarily integers) in every line of input and terminated by EOF (end of file)."
Then you should write

```
double a,b,c;
while(scanf("%lf %lf %lf",&a,&b,&c)==3) //three inputs are taken correctly
{
// rest of the work
}
```

Example 4:
If the problem states,
"There will be 3 integers and one name (not consisting of any space) and the input will be terminated if the given first integer is 0"
Then you should write

```
int a,b,c;
char name[100];
while(scanf("%d %d %d %s",&a,&b,&c,name)==4)
{
if(a==0)
break;
// rest of the work
}
```

Example 5:
What would be the output of the following code?
char name[100]; input is given as:
scanf("%s", name); String input
printf("%s", name);
The output will be "String" but not "String input". Because scanf() function takes string input without space. To take the whole line you can use gets() function or modified scanf() function. For example:
scanf("%[^\n]",name);
This will take the input correctly.
gets:
Suppose you have to take a line which contains a name, or names but can have spaces then you have to use gets.

Example 6:
Return value of gets() function:
a. gets() function returns the pointer of the argument on success.

b. On error, gets() function returns -1 or NULL. The error will occur because of end of file (EOF)
For example:
char line[1001], *pt;
pt=gets(line);
printf("%s\n",pt);
For the correct input, pt points line[0] and prints the output that is given as input.
For the incorrect input (such as EOF), pt will be -1 and prints "**(null)"** as output.

Example 7:
Suppose the problem states,
"There will be one or more names in a single line and will be terminated
by end of file"
Then you should write
char line[1001];
while(gets(line))
{
// rest of the work
}

Input & output from file:

int main()
{
freopen("input.txt","r",stdin); // note : Be carefull when run the code we need to change "r" by the "w".
freopen("output.txt","w",stdout);

// Your code
return 0;
}

Number Theory

Sieve of Eratosthenes (Prime Numbers):

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

int nprime,n=10000003;
vector<int>prime;
int mark[10000004];

void prim()
{
   int limit=sqrt((n*1.0)+2);
   mark[1]=0;

   for(int i=4; i<=n; i+=2)
   {
      mark[i]=1;
   }

   prime.push_back(2);

   for(int i=3; i<=n; i++)
   {
      if(!mark[i])
      {
         prime.push_back(i);
         if(i<limit)
         {
            for(int j=i*i; j<=n; j+=i*2)
            {
               mark[j]=1;
            }
         }
      }
   }
}
```

```
int main()
{
   prim();
   for(int i=0; i<100; i++)
   {
      cout<<prime[i]<<endl;
   }
}
```

Ref: Mahabul Hasan book, page:52

Memory Efficient Sieve (Prime Numbers):

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

typedef long long int ll;
typedef unsigned long long ull;


#define sz 10000009
bitset<10000013>bs;
vector<int>prime;

void seive()
{
   bs.set(); // set all bits to 1

   bs[0]=0;
   bs[1]=0;

   for(ll i=2; i<sz; i++)
   {
      if(bs[i]==1)
      {
         for(ll j=i*i; j<sz; j+=i)
         {
            bs[j]=0;
         }
```

```
            prime.push_back((int)i);
        }
    }
}

bool isprime(ll N)
{
    if(N<=sz)
    {
        return bs[N];
    }
    for(int i=0; i<prime.size(); i++)
    {
        if(N%prime[i]==0)
        {
            return false;
        }
        return true;
    }
}

int main()
{
    seive();
    printf("%d\n",isprime(2147483647));
}
```

Ref: Cp3 book,page:211

Segment Sieve(prime number):

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

#define setz(a) memset(a, 0, sizeof(a))
typedef unsigned long long uLL;

vector<int>prime;
int mark[1000002];
int arr[100005];
```

```
void sieve(int n)
{
   int i,j,limit=sqrt(n*1.0)+2;
   mark[1]=1;
   for(int i=4; i<=n; i+=2)
   {
      mark[i]=1;
   }
   prime.push_back(2);
   for(int i=3; i<=n; i+=2)
   {
      if(!mark[i])
      {
         prime.push_back(i);
         if(i<=limit)
         {
            for(int j=i*i; j<=n; j+=i*2)
            {
               mark[j]=1;
            }
         }
      }
   }
}

void segment(int a,int b)
{
   if(a==1)
      a++;
   int srqtn=sqrt(b);
   int j,p;
   setz(arr);
   for(int i=0; i<prime.size() && prime[i]<=srqtn; i++)
   {
      p=prime[i];
      j=p*p;
      if(j<a)
         j=((a+p-1)/p)*p;
      for(int k=j; k<=b; k+=p)
```

```
        {
            arr[k-a]=1;
        }

    }

    for(int i=a; i<=b; i++)
    {
        if(arr[i-a]==0)
        {
            printf("%d\n",i);
        }
    }
}

int main()
{
    sieve(100005);
    int t,x,y;
    cin>>t;
    while(t--)
    {
        cin>>x>>y;
        segment(x,y);
    }
}
```

Efficient Algo Finding Divisor:

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

typedef unsigned long long uLL;
vector<int>d;

int main()
{
    int n,s;
    cin>>n;
    int temp=sqrt(n);
```

```
    for(int i=1; i<=temp; i++)
    {
        if(n%i==0)
        {
            s=n/i;
            if(s!=i)
            {
                d.push_back(i);
                d.push_back(s);
            }
            else
            {
                d.push_back(s);
            }
        }
    }
    for(int i=0; i<d.size(); i++)
    {
        cout<<d[i]<<" ";
    }
}
```

Prime Factorization:

```
#include<bits/stdc++.h>
#define N 10000000

using namespace std;
int mark[10000003];

map<int,int>m;
map<int,int>:: iterator it;
vector<int>p;

void prime()
{
    int limit=sqrt(N)+2;
    for(int i=4; i<=N; i+=2)
    {
```

```
        mark[i]=1;
    }
    p.push_back(2);
    for(int i=3; i<=N; i+=2)
    {
        if(mark[i]==0)
        {
            p.push_back(i);
            if(i<=limit)
            {
                for(int j=i*i; j<=N; j+=i)
                {
                    mark[j]=1;
                }
            }
        }
    }
}

void primefactor(int n)
{
    int temp=(int)sqrt(n);
    for(int i=0; p[i]<=temp && i<p.size(); i++)
    {
        if(n%p[i]==0)
        {
            while(n%p[i]==0)
            {
                n/=p[i];
                m[p[i]]++;
            }
            temp=sqrt(n); /// for reduce time complexity
        }
    }
    if(n>1)
    {
        m[n]=1;
    }
```

```
}
int main()
{
   prime();
   int n;
   cin>>n;
   primefactor(n);
   for(it=m.begin(); it!=m.end(); it++)
   {
      cout<<it->first<<"^"<<it->second<<endl;
   }
}
```

Finding number of Divisor By Prime Factorization:

Formula: (e1+1)*(e2+1)…(e^n+1).

Example: prime factor of 18 is = 2^1 * 3^2
=(p1^ e1 * p2^e2)
Here e1=1 and e2=2;
So, Ans is =(1+1)*(2+1);
=6;

```
#include<bits/stdc++.h>
#define N 10000000

using namespace std;
int mark[10000003];
map<int,int>m;
map<int,int>:: iterator it;
vector<int>p;

void prime()
{
   int limit=sqrt(N)+2;
   for(int i=4; i<=N; i+=2)
   {
      mark[i]=1;
   }
   p.push_back(2);
   for(int i=3; i<=N; i+=2)
```

```
    {
        if(mark[i]==0)
        {
            p.push_back(i);
            if(i<=limit)
            {
                for(int j=i*i; j<=N; j+=i)
                {
                    mark[j]=1;
                }
            }
        }
    }
}

void primefactor(int n)
{
    int temp=(int)sqrt(n);
    for(int i=0; p[i]<=temp && i<p.size(); i++)
    {
        if(n%p[i]==0)
        {
            while(n%p[i]==0)
            {
                n/=p[i];
                m[p[i]]++;
            }
        }
    }
    if(n>1)
    {
        m[n]=1;
    }


}

int main()
{
    prime();
```

```
    int n,divisor=1;
    cin>>n;
    primefactor(n);
    for(it=m.begin(); it!=m.end(); it++)
    {
       divisor*=(it->second)+1;
    }
    cout<<"Number of divisor  "<<n<<" is "<<divisor<<endl;
}
```

Finding Summation of Divisor By Prime Factorization:

Formula: $\frac{p1^{e1+1}-1}{p1-1} * \frac{p2^{e2+1}-1}{p2-1} * \frac{p3^{e2+1}-1}{p3-1}$

Example: prime factor of 18 is = 2^1 * 3^2
= (p1^ e1 * p2^e2)

Here, p1=2 , p2=3 and e1=1, e2=2;

$$= \frac{2^{1+1}-1}{2-1} * \frac{3^{2+1}-1}{3-1}$$

So, summation divisor 18 is = 39.

```
#include<bits/stdc++.h>
#define N 10000000

using namespace std;
int mark[10000003];

map<int,int>m;
map<int,int>:: iterator it;
vector<int>p;

void prime()
{
    int limit=sqrt(N)+2;
    for(int i=4; i<=N; i+=2)
    {
```

```
        mark[i]=1;
    }
    p.push_back(2);
    for(int i=3; i<=N; i+=2)
    {
        if(mark[i]==0)
        {
            p.push_back(i);
            if(i<=limit)
            {
                for(int j=i*i; j<=N; j+=i)
                {
                    mark[j]=1;
                }
            }
        }
    }
}

void primefactor(int n)
{
    int temp=(int)sqrt(n);
    for(int i=0; p[i]<=temp && i<p.size(); i++)
    {
        if(n%p[i]==0)
        {
            while(n%p[i]==0)
            {
                n/=p[i];
                m[p[i]]++;
            }
        }
    }
    if(n>1)
    {
        m[n]=1;
    }


}
```

```
int main()
{
   prime();
   int n,sum=1,r;
   cin>>n;
   primefactor(n);
   for(it=m.begin(); it!=m.end(); it++)
   {
      r=(round(pow(it->first,it->second+1))-1)/(it->first-1);
      sum*=r;

   }
   cout<<"summation of divisor  "<<n<<" is "<<sum<<endl;
}
```

Finding Summation even of Divisor By Prime Factorization:

Formula :

Step 1:First of all we find the Prime factorization of the given number.
example: 720=2^4 * 3^2 * 5^1.

Step 2: Without prime factor 2 we write a equation summation of power 0 to highest power of that prime factor.When prime factor is 2 we start this method from 1 to highest power of prime factor 2.Let's see this example for clearly understand.

$720=(2^1+2^2+2^3+2^4)*(3^0+3^1+3^2)*(5^0+5^1)$
=30*13*16
=2340.

```
#include<bits/stdc++.h>
#define N 10000000
using namespace std;

int mark[10000003];
```

```
map<int,int>m;
map<int,int>:: iterator it;
vector<int>p;

void prime()
{
   int limit=sqrt(N)+2;
   for(int i=4; i<=N; i+=2)
   {
      mark[i]=1;
   }
   p.push_back(2);
   for(int i=3; i<=N; i+=2)
   {
      if(mark[i]==0)
      {
         p.push_back(i);
         if(i<=limit)
         {
            for(int j=i*i; j<=N; j+=i)
            {
               mark[j]=1;
            }
         }
      }
   }
}

void primefactor(int n)
{
   int temp=(int)sqrt(n);
   for(int i=0; p[i]<=temp; i++)
   {
      if(n%p[i]==0)
      {
         while(n%p[i]==0)
         {
            n/=p[i];
            m[p[i]]++;
         }
```

```
        }
    }
    if(n>1)
    {
        m[n]=1;
    }


}

int main()
{
    prime();
    int n,sum=1,r;
    cin>>n;
    primefactor(n);
    for(it=m.begin(); it!=m.end(); it++)
    {
        if(it->first==2)
        {
            int temp=0;
            for(int i=1; i<=it->second; i++)
            {
                temp+=round(pow(it->first,i));
            }
            sum*=temp;
        }
        else
        {
            int temp=0;
            for(int i=0; i<=it->second; i++)
            {
                temp+=round(pow(it->first,i));
            }
            sum*=temp;
        }
    }
    cout<<"summation of odd divisor  "<<n<<" is "<<sum<<endl;
}
```

Finding Summation Odd of Divisor By Prime Factorization:

Formula :

Step 1:First of all we neet to find the Prime factorization of the given number.
example: 720=2^4 * 3^2 * 5^1.

Step 2: Without prime factor 2 we write a equation summation of power 0 to highest power of that prime factor.When prime factor is 2 we use only 2^0.Let's see this example for clearly understand.

$720=(2^0) *(3^0 + 3^1 + 3^2) * (5^0 + 5^1)$
=1*13*6
=78.

```
#include<bits/stdc++.h>
#define N 10000000
using namespace std;

int mark[10000003];
map<int,int>m;
map<int,int>:: iterator it;
vector<int>p;

void prime()
{
   int limit=sqrt(N)+2;
   for(int i=4; i<=N; i+=2)
   {
      mark[i]=1;
   }
   p.push_back(2);
   for(int i=3; i<=N; i+=2)
   {
      if(mark[i]==0)
      {
         p.push_back(i);
         if(i<=limit)
         {
```

```
            for(int j=i*i; j<=N; j+=i)
            {
               mark[j]=1;
            }
         }
      }
   }
}

void primefactor(int n)
{
   int temp=(int)sqrt(n);
   for(int i=0; p[i]<=temp; i++)
   {
      if(n%p[i]==0)
      {
         while(n%p[i]==0)
         {
            n/=p[i];
            m[p[i]]++;
         }
      }
   }
   if(n>1)
   {
      m[n]=1;
   }

}

int main()
{
   prime();
   int n,sum=1,r;
   cin>>n;
   primefactor(n);
   for(it=m.begin(); it!=m.end(); it++)
   {
      if(it->first==2)
      {
```

```
            sum*=1;
        }
        else
        {
            int temp=0;
            for(int i=0; i<=it->second; i++)
            {
                temp+=round(pow(it->first,i));
            }
            sum*=temp;
        }
    }
    cout<<"summation of odd divisor  "<<n<<" is "<<sum<<endl;
}
```

A brute force efficient method finding summation of even divisor:

Note: For better understand this code please write step by step in a paper.

```
#include<bits/stdc++.h>
using namespace std;
typedef long long LL;
typedef unsigned long long uLL;

LL a[1000];
int main()
{
    LL sum;
    int n;
    for(int i=2; i<=100; i+=2)
    {
        sum=0;
        for(int j=i; j<=100; j+=i)
        {
            a[j]+=i;
        }
    }
    for(int i=1; i<20; i++)
    {
        cout<<a[i]<<endl;
    }
}
```

Eular's Totient Funtion :

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

int phi(int n)
{
   int ret=n;
   for(int i=2; i*i<=n; i++)
   {
      if(n%i==0)
      {
         while(n%i==0)
         {
            n/=i;
         }
         ret=ret-ret/i;
      }
   }
   if(n>1)
   {
      ret=ret-ret/n;
   }
   return ret;
}

int main()
{
   int n,ans;
   cin>>n;
   ans=phi(n);
   cout<<"Eular phi "<<ans<<endl;
}
```

GCD (greatest common divisor):

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

```
#define ll long long int

ll gcd(ll x,ll y)
{
   ll mod;
   while(x%y)
   {
      mod = x%y;
      x = y;
      y = mod;
   }
   return y;
}

int main()
{
  cout<<gcd(10,12)<<endl;
}
```

LCM of n elements:

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

typedef long long int ll;
int arr[] = {2, 7, 3, 9, 4};

int gcd(int a, int b)
{
   if (b==0)
      return a;
   return gcd(b, a%b);
}

ll findlcm(int n)
{

   ll ans = arr[0];
```

```
    for (int i=1; i<n; i++)
        ans = ( ((arr[i]*ans)) / (gcd(arr[i], ans)) );

    return ans;
}


int main()
{
    printf("%lld", findlcm(5));
    return 0;
}
```

Big mod:

Find 2^1000 mod 11 :

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long int
#define M 11

ll F(ll N,ll P)
{
    ll ret;
        if(P==0) return 1;
        if(P%2==0)
        {
                ll ret=F(N,P/2);
                return ((ret%M)*(ret%M))%M;
        }
        else {
      ret=((N%M)*(F(N,P-1)%M))%M;
      return ret;
        }

}


int main()
```

```
{
cout<<F(2,1000)<<endl;
}
```

Combination with Modular Inverse (when mod value is a Prime Number):

Problem link: http://www.lightoj.com/volume_showproblem.php?problem=1067

Find: C_r^n or $\frac{n!}{r!(n-r)!}$ mod 1000000007

We can not do modular division generally.If mod value is prime then we do Modular inverse write M-2.

```
#include<bits/stdc++.h>
using namespace std;
#define uLL unsigned long long int
#define M 1000000007
uLL f[1006];

void fact()
{
   uLL j,res,i;
   f[0]=1;
   for(j=1;j<=1006;j++){
      f[j]=((f[j-1]%M)*(j%M))%M;
   }
}

uLL modular_inverse(uLL a,uLL b)
{
   uLL ret;
   if(b==0)
      return 1;
   if(b%2==0)
   {
      ret=modular_inverse(a,b/2);
      return ((ret%M)*(ret%M))%M;
   }
   else
```

```
    return ((a%M)*(modular_inverse(a,b-1)%M))%M;
}

int main()
{
   int t,n,r,k;
   uLL a,b,c,temp,x,y,q;
   fact();
   scanf("%d",&t);
   for(int i=1; i<=t; i++)
   {
      scanf("%d%d",&n,&r);
      k=n-r;
      a= f[n];
      y=modular_inverse(n+1,M-2);
      x=modular_inverse(f[r],M-2);
      q=((y%M)*(x%M))%M;
      temp=((a%M)*(q%M))%M;
      printf("Case %d: %llu\n",i,temp);
   }
}
```

Divisibility Rules:

Rules for Zero: Divisible by all integer, (expect zero)
Rules for Two : If last digit must Divisible by 2 (ex: 12)
Rules for three: If sum of digit is divisible by 3. (ex 123) , 1+2+3 = 6 (6 % 3 = 0)
Rules for Four: If last two digits divisible by 4(ex 1231224) last 2 digits are 24 (24 % 4 == 0)
Rules for Five: If last digit is 0 or 5.
Rules for Six: if the number is divisible by 2 & 3
Rules for Eight: If last 3 digits is divisible by 8
Rules for Nine: If sum of digit is divisible by 9.
Rules for Twelve: If the number is divisible by 3 & 4.
Rules for Eighteen: 18 = 2 * 9 check (last digit is divisible 2 and sum of digits is divisible by nine)
Rules for Thirty Six : 36 = (4 * 9) check last two digits are divisible by 4 and sum of digits is divisible by nine

Graph Theory

Bfs Code:

Tutorial link: http://www.shafaetsplanet.com/planetcoding/?p=604

```
#include<bits/stdc++.h>
using namespace std;
vector<int>G[100];
int visited[100],level[100];
int parent[100];


void bfs(int n,int src)
{
   queue<int>Q;
   Q.push(src);
   visited[src]=1;
   level[src]=0;
   while(!Q.empty())
   {
      int u=Q.front();
      for(int i=0; i<G[u].size(); i++)
      {
         int v=G[u][i];
         if(!visited[v])
         {
            level[v]=level[u]+1;
            parent[v]=u;
            visited[v]=1;
            Q.push(v);
         }
      }
      Q.pop();
   }
   for(int i=1; i<=n; i++)
   {
      printf("%d to %d distance %d",src,i,level[i]);
   }
```

```
}

/// first we find Adjacency list of the give graph.Then we take input total number of node and edge.
int main()
{
   int node,edge,x,y,s;
   cin>>node>>edge; /// input total number of node and edge.
   for(int i=1; i<=edge; i++) ///inout adjacency node of graph.
   {
      cin>>x>>y;
      G[x].push_back(y);
      G[y].push_back(x);
   }
   cin>>s;/// source input
   bfs(s,node); /// Here start bfs from node 6.

}
```

Bfs on 2D grid:

Tutorial link: http://www.shafaetsplanet.com/planetcoding/?p=604

```
#define pii pair<int,int>
int fx[]= {1,-1,0,0}; //ডিরেকশন অ্যারে
int fy[]= {0,0,1,-1};
int cell[100][100]; //cell[x][y] যদি -১ হয় তাহলে সেলটা ব্লক
int d[100][100],vis[100][100]; //d means destination from source.
int row,col;

void bfs(int sx,int sy) //Source node is in [sx][sy] cell.
{
   memset(vis,0,sizeof vis);
   vis[sx][sy]=1;
   queue<pii>q; //A queue containing STL pairs
   q.push(pii(sx,sy));
   while(!q.empty())
   {
      pii top=q.front();
```

```
        q.pop();
        for(int k=0; k<4; k++)
        {
            int tx=top.first+fx[k];
            int ty=top.second+fy[k]; //Neighbor cell [tx][ty]
 if(tx>=0 and tx<row and ty>=0 and ty<col and cell[tx][ty]!= -1 and
vis[tx][ty]==0) //Check if the neighbor is valid and not visited before.
            {
                vis[tx][ty]=1;
                d[tx][ty]=d[top.first][top.second]+1;
                q.push(pii(tx,ty)); //Pushing a new pair in the queue
            }
        }
    }
}
```

Dijkstra's algorithm :

```
#include<bits/stdc++.h>
using namespace std;
#define pii pair<int,int>

vector<pii>G[1000];
int d[1000],parent[1000];
int n,e,w;

int path(int j)
{
    if(parent[j]==-1)
        return 0;
    path(parent[j]);
    printf("%d ",j);
}

void dijktra(int s)
{
    for(int i=1; i<=n; i++)
    {
        d[i]=999999999;
    }
```

```
    d[s]=0;
    parent[s]=-1;
    int u,c,v,wt;
    priority_queue<pii,vector<pii>,greater<pii> >q;
    q.push(pii(s,0));
    while(!q.empty())
    {
        u=q.top().first;
        c=q.top().second;
        q.pop();
        if(d[u]<c)
            continue;
        for(int i=0; i<G[u].size(); i++)
        {
            v=G[u][i].first;
            wt=G[u][i].second;
            if(d[v]>d[u]+wt)
            {
                d[v]=d[u]+wt;
                parent[v]=u;
                q.push(pii(v,d[v]));
            }
        }
    }
}

int main()
{
    int x,y;
    cin>>n>>e;
    for(int i=1; i<=e; i++)
    {
        cin>>x>>y>>w;
        G[x].push_back(pii(y,w));
        G[y].push_back(pii(x,w));
    }
    dijktra(1);
    for(int i=1; i<=n; i++)
    {
        if(i==1)
```

```
            continue;
        else if(d[i]>=999999999)
            cout<<"not rech from source"<<endl;
        else
            cout<<"shorest distent for node"<<i<<" is "<<d[i]<<endl;
    }
    path(n);
}
```

Bellman Ford algorithm:

```
#include <bits/stdc++.h>
using namespace std;
#define pii pair<int,int>
int d[1200];
vector<pii>adj[1100];
int n,e,x,y,w,res;

void init()
{
    for(int i=0; i<n; i++)
    {
        d[i]=99999999999;
    }
    d[0]=0;
}

int bell()
{
    init();
    int v,wt;
    for(int i=0; i<n; i++)
    {
        for(int u=0; u<n; u++)
        {
            for(int j=0; j<adj[u].size(); j++)
            {
                v=adj[u][j].first;
                wt=adj[u][j].second;
                if(i==n-1 && d[v]>d[u]+wt)
```

```
            {
                return 0;
            }
            else if(d[v]>d[u]+wt)
            {
                d[v]=d[u]+wt;
            }
        }
    }
}
return 1;
}

int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        cin>>n>>e;
        for(int i=1; i<=e; i++)
        {
            cin>>x>>y>>w;
            adj[x].push_back(pii(y,w));
        }
        res=bell();
        if(res)
        {
            cout<<"not possible"<<endl;
        }
        else
            cout<<"possible"<<endl;
        for(int i=0;i<n;i++){
            adj[i].clear();
        }
    }

}
```

Floyd warshall algorithm :

```
#include<bits/stdc++.h>
using namespace std;
int d[100][100],s[100][100];
int n,e,w;

void folyed(){
for(int k=1;k<=n;k++){
   for(int i=1;i<=n;i++){
      for(int j=1;j<=n;j++){
         if(d[i][j]>d[i][k]+d[k][j]){
            d[i][j]=d[i][k]+d[k][j];
            s[i][j]=s[k][j];
         }
      }
   }
}
}

int main(){
int x,y;
cin>>n>>e;
for(int i=1;i<=n;i++){
   for(int j=1;j<=n;j++){
      d[i][j]=99999999;
      s[i][j]=-1;
      if(i==j)
         d[i][j]=0;
   }
}
for(int i=1;i<=e;i++){
   cin>>x>>y>>w;
   d[x][y]=w;
   d[y][x]=w;
   s[x][y]=x;
   s[y][x]=y;
}

folyed();
```

```
cout<<"D table "<<endl;
for(int i=1;i<=n;i++){
   for(int j=1;j<=n;j++){
      cout<<d[i][j]<<" ";
   }
   cout<<endl;
}
cout<<endl;

cout<<"S table : "<<endl;
for(int i=1;i<=n;i++){
   for(int j=1;j<=n;j++){
      cout<<s[i][j]<<" ";
   }
   cout<<endl;
}
}
```

DFS Algorithm:

```
#include<bits/stdc++.h>
using namespace std;
vector<int>adj[100];
int visited[100];
void dfs(int u){
int v;
visited[u]=1;
for(int j=0;j<adj[u].size();j++){
   v=adj[u][j];
   if(visited[v]==0){
      cout<<v<<" ";
      dfs(v);
   }
}
}
int main(){
int edge,node,x,y;
cin>>edge>>node;
for(int i=1;i<=edge;i++){
```

```
    cin>>x>>y;
    adj[x].push_back(y);
    adj[y].push_back(x);
}
cout<<1<<" ";
dfs(1);
}
```

Cycle Finding Directed graph By DFS:

```
#include<bits/stdc++.h>
using namespace std;
vector<int>adj[1000];
int visited[100];
int n,e,v,flag=0;

void dfs(int u){
visited[u]=1;
if(flag==1)
    return;
for(int i=0;i<adj[u].size();i++){
    int v=adj[u][i];
    if(visited[v]==1){
        flag=1;
        return;
    }
    else if(visited[v]==0){
        dfs(v);
    }
}
visited[u]=2;
}

int main(){
cin>>n>>e;
int cn=0,x,y;
for(int i=0;i<e;i++){
    cin>>x>>y;
    adj[x].push_back(y);
```

```
}
dfs(1);
if(flag==1){
   cout<<"Cycle found"<<endl;
}
else{
   cout<<"Cycle not found"<<endl;
}

}
```

Cycle Finding Undirected graph By DFS:

```
#include<bits/stdc++.h>
using namespace std;
vector<int>adj[1000];
int visited[100];
int n,e,v,flag=0;
int s;


int dfs(int u,int node)
{
   visited[u]=1;
   if(flag==1)
      return 0;
   for(int j=0; j<adj[u].size(); j++)
   {
      v=adj[u][j];
      if(visited[v]==1 && v!=node)
      {
         flag=1;
         return 0;
      }
      else if(visited[v]==0)
      {
         dfs(v,u);
      }
   }
```

```
}

int main()
{
   cin>>n>>e;
   int cn=0,x,y;
   for(int i=0; i<e; i++)
   {
      cin>>x>>y;
      adj[x].push_back(y);
      adj[y].push_back(x);
   }
   dfs(1,-1);
   if(flag==1)
   {
      cout<<"Cycle found"<<endl;
   }
   else
   {
      cout<<"Cycle not found"<<endl;
   }


}
```

Topological Sort :

Condition for Topological sort:

- Graph need to directed graph
- Graph have not any cycle.

Problem : uva 11686

```
#include <bits/stdc++.h>

using namespace std;
#define pb(a) push_back(a)
#define setz(a) memset(a, 0, sizeof(a))
#define isc(a) scanf("%d", &a)
#define isc2(a,b) scanf("%d%d", &a,&b)
```

```
typedef long long LL;
typedef unsigned long long uLL;

int visited[1000002],s,flag;
vector<int>vs;
vector<int>adj[1000002];

void dfs(int u)
{
   visited[u]=1;
   if(flag==1)
      return;

   for(int i=0; i<adj[u].size(); i++)
   {
      int v=adj[u][i];
      if(visited[v]==1)
      {
         flag=1;
         return;
      }
      else if(visited[v]==0)
      {
         dfs(v);
      }
   }
   visited[u]=2;
   vs.pb(u);
}

int main()
{
   int n,e,x,y;
   while(cin>>n>>e)
   {
      if(n==0 && e==0)
         break;
      setz(visited);
      vs.clear();
      while(e--)
```

```
        {
            isc2(x,y);
            adj[x].pb(y);
        }
        flag=0;
        dfs(1);
        if(flag==1)
            cout<<"IMPOSSIBLE"<<endl;
        else
        {
            vs.clear();
            setz(visited);
            for(int i=1; i<=n; i++)
            {
                if(!visited[i])
                {
                    dfs(i);
                }
            }
            for(int i=vs.size()-1; i>=0; i--)
            {
                printf("%d\n",vs[i]);
            }
        }
        for(int i=1; i<=n; i++)
        {
            adj[i].clear();
        }
    }
}
```

Strongly connected component :

```
#include<bits/stdc++.h>
using namespace std;
int visited[1000];
vector<int>adj[1000];
vector<int>transpose[10000];
vector<int>vs;
```

```
void dfs(int u)
{
   visited[u]=1;
   for(int i=0; i<adj[u].size(); i++)
   {
      int v=adj[u][i];
      if(visited[v]==0)
      {
         dfs(v);
      }
   }
   vs.push_back(u);
}
void dfs2(int u)
{
   visited[u]=1;
   for(int i=0; i<transpose[u].size(); i++)
   {
      int v=transpose[u][i];
      if(visited[v]==0)
      {
         cout<<v;
         dfs2(v);
      }
   }
}

int main()
{
   int n,e,x,y,vs[10000];
   while(cin>>n>>e)
   {
      string x,y;
      int k=1;
      for(int i=1; i<=e; i++)
      {
         cin>>x>>y;
      }
   }
   for(int i=1; i<=e; i++)
```

```
    {
        cin>>x>>y;
        adj[x].push_back(y);
        transpose[y].push_back(x);
    }
    dfs(1);
    memset(visited,0,sizeof(visited));
    for(int i=vs.size()-1; i>=0; i--)
        if(!visited[vs[i]])
        {
            cout<<"ss: "<<vs[i];
            dfs2(vs[i]);
            cout<<endl;
        }
}
```

Articulation point :

```
#include <bits/stdc++.h>
#define pb(a) push_back(a)
#define setz(a) memset(a, 0, sizeof(a))
using namespace std;
vector<int>adj[10005];
int
visited[10005],start_time[10005],low_time[10005],parent[10005],tim=0,child_cou
nt=0,v,root,total;

void Articulationpoin(int u)
{
    visited[u]=1;
    start_time[u]=tim;
    low_time[u]=tim;
    tim++;
    bool isArti =false;
    for(int i=0; i<adj[u].size(); i++)
    {
        v=adj[u][i];   /// adj of each node u
        if(v==parent[u])
            continue;
        if(!visited[v])     /// if unvisited tree edge
```

```
        {
            if(u==root)child_count++;
            parent[v]=u;
            Articulationpoin(v);

            int k=adj[u][i];      /// it's important adj[u][i] save again new variable
            if(start_time[u]<=low_time[k])   ///check condition start_time(u)<=
low_time(v)
                isArti=true;
            else
                low_time[u]=min(low_time[u],low_time[k]); ///
low_time[u]=min(low_time[u],low_time[v]

        }

        else
        {
            low_time[u]=min(low_time[u],start_time[v]);  /// else u is visited that
means back edge then low_time[u]=min(low_time[u],start_time[v])
        }
    }
    if((parent[u]==-1 && child_count>=2)||(parent[u]!=-1 && isArti)) /// for count
Articulation point
    {
        total++;
    }
}

int main()
{
    int node,edge,x,y,t;
    cin>>t;
    for(int j=1; j<=t; j++)
    {
        cin>>node>>edge;
        setz(visited);
        setz(parent);
        setz(start_time);
        setz(low_time);
        total=0;
```

```
        tim=0;
        for(int k=1; k<=edge; k++)
        {
            cin>>x>>y;
            adj[x].pb(y);
            adj[y].pb(x);
        }
        for(int i=1; i<=node; i++)
        {
            if(!visited[i])
            {
                 child_count=0;
                parent[i]=-1; /// for root node assing null
                root=i;         /// save root node
                Articulationpoin(i); /// call dfs
            }
        }
        printf("Case %d: ",j);
        cout<<total<<endl;
        for(int i=1; i<=node; i++)
        {
            for(int j=0; j<adj[i].size(); j++)
            {
                adj[i].clear();
            }
        }
    }
}
```

Data Structure

Disjoinset

Tutorial link :

- **http://www.shafaetsplanet.com/planetcoding/?p=763**
- **https://www.youtube.com/watch?v=ID00PMy0-vE&spfreload=10**

Problem: Uva 10685

Find the the size of the largest group:

```
#include<bits/stdc++.h>
using namespace std;
int parent[100004],cn[100004],n,visited[100004];
map<string,int>mp;
map<string,int>:: iterator it;

int find_frnd(int x)
{
   if(parent[x]==x)
      return x;
   else
      return parent[x]=find_frnd(parent[x]);

}

void uni(int a,int b)
{
   cn[b]+=cn[a];
   parent[a]=parent[b];
   printf("%d\n",cn[b]);
}

int main()
{

   int t,v;
```

```
string x,y;
cin>>t;
while(t--)
{
    mp.clear();
     memset(cn,0,sizeof(cn));
     memset(parent,0,sizeof(parent));
     memset(visited,0,sizeof(visited));
    cin>>n;
    int k=1;
    for(int j=1; j<=n; j++)
    {
        int flag1=0,flag2=0;
        cin>>x;
        cin>>y;
        if(visited[mp[x]]==0){
         mp[x]=k;
         parent[k]=k;
         cn[k]=1;
         k++;
         visited[mp[x]]=1;
        }
         if(visited[mp[y]]==0){
            mp[y]=k;
            parent[k]=k;
            cn[k]=1;
            k++;
            visited[mp[y]]=1;
         }
        int pa=find_frnd(mp[x]);
        int pb=find_frnd(mp[y]);
        if(pa != pb)
        {
            uni(pa, pb);
        }
        else{
            printf("%d\n",cn[pa]);
        }
    }
```

```
    }

}
```

Cycle find in undirected graph by disjoin set:

```
#include <bits/stdc++.h>
using namespace std;
int n;
int parent[1005];

void built (int n)
{
    for(int i = 1; i <= n; i++)
    {
        parent[i] = -1;
    }
}

int find_parent(int n)
{
    if(parent[n] == -1) return n;
    else return parent[n] = find_parent(parent[n]);
}

void uni(int a,int b)
{
    parent[a]=b;
}

int main()
{
    int m,flag=0;
    cin >> n >> m;   /// number of nodes and edge
    built(n);
    for(int i = 0 ; i < m; i++)
    {
        int a, b;
        cin>>a>>b;
        int pa = find_parent(a);
```

```
        int pb = find_parent(b);
        if(pa != pb)
        {
          uni(pa, pb);
        }
        else{                  /// if pa == pb then cycle find
          flag=1;
        }
    }
    if(flag==1)
        cout<<"cycle found "<<endl;
    else
        cout<<"No cycle have "<<endl;
}
```

Kruskal Algorithm for finding minimum spanning tree:

```
#include<bits/stdc++.h>
using namespace std;
const int MAX = 100005;
int id[MAX], nodes, edges;
pair <long long, pair<int, int> > p[MAX];

void initialize()
{
    for(int i = 1; i <= MAX; ++i)
        id[i] = i;
}

int root(int x)
{
    while(id[x] != x)
    {
        id[x] = id[id[x]];
        x = id[x];
    }
    return x;
}

void union1(int x, int y)
```

```
{
   int p = root(x);
   int q = root(y);
   id[p] = id[q];
}

long long kruskal(pair<long long, pair<int, int> > p[])
{
   int x, y;
   long long cost, minimumCost = 0;
   for(int i = 0; i < edges; ++i)
   {
      // Selecting edges one by one in increasing order from the beginning
      x = p[i].second.first;
      y = p[i].second.second;
      cost = p[i].first;
      // Check if the selected edge is creating a cycle or not
      if(root(x) != root(y))
      {
         minimumCost += cost;
         union1(x, y);
      }
   }
   return minimumCost;
}

/// It's needed when finding maximumspanning tree
bool cmp(int i,int j)
{
   return i>j;
}

int main()
{
   int x, y;
   long long weight, cost, minimumCost;
   initialize();
   cin >> nodes >> edges;
   for(int i = 0; i < edges; ++i)
   {
```

```
        cin >> x >> y >> weight;
        p[i] = make_pair(weight, make_pair(x, y));
    }
    // Sort the edges in the ascending order
    sort(p, p + edges);
    minimumCost = kruskal(p);
    cout << minimumCost << endl;
    return 0;
}
```

Segment tree

Find sum index i to index j in a given array:

```
#include<bits/stdc++.h>
#define mx 100001

using namespace std;
int tree[mx*3],arr[mx];

void init(int node,int b,int e)
{
    if(b==e)
    {
        tree[node]=arr[b];
        return;
    }
    int left=node*2;
    int right=(node*2)+1;
    int mid=(b+e)/2;
    init(left,b,mid);
    init(right,mid+1,e);
    tree[node]=tree[left]+tree[right];
}

int query(int node,int b,int e,int i,int j)
{
    if(i>e || j<b)
    {
        return 0;
```

```
    }
    if(b>=i && e<=j)
    {
        return tree[node];
    }
    int left=node*2;
    int right=(node*2)+1;
    int mid=(b+e)/2;
    int p1=query(left,b,mid,i,j);
    int p2=query(right,mid+1,e,i,j);
    return p1+p2;
}


void update(int node,int b,int e,int i,int newvalu)
{
    if(i>e || i<b)
    {
        return ;
    }
    if(b>=i && e<=i)
    {
        tree[node]=newvalu;
        return;
    }
    int left=node*2;
    int right=(node*2)+1;
    int mid=(b+e)/2;
    update(left,b,mid,i,newvalu);
    update(right,mid+1,e,i,newvalu);
    tree[node]=tree[left]+tree[right];
}


int main()
{
    int n;
    cin>>n;
    for(int i=1; i<=n; i++)
    {
```

```
        cin>>arr[i];
    }
    init(1,1,n);
    cout << query(1, 1, n, 2, 4) << endl;
    update(1,1,n,2,1);
    cout << query(1, 1, n, 2, 4) << endl;
}
```

Find minimum value index i to index j in a given array (RMQ) :

```
#include<bits/stdc++.h>
#define mx 100009
using namespace std;
int tree[mx*3],arr[mx];

void init(int node,int b,int e)
{
    if(b==e)
    {
        tree[node]=arr[b];
        return;
    }
    int left=node*2;
    int right=(node*2)+1;
    int mid=(b+e)/2;
    init(left,b,mid);
    init(right,mid+1,e);
    if(tree[left]>tree[right])
    {
        tree[node]=tree[right];
    }
    else
        tree[node]= tree[left];
}

int query(int node,int b,int e,int i,int j)
{
    if(i>e || j<b)
    {
        return 100000000;
```

```
    }
    if(b>=i && e<=j)
    {
        return tree[node];
    }
    int left=node*2;
    int right=(node*2)+1;
    int mid=(b+e)/2;
    int p1=query(left,b,mid,i,j);
    int p2=query(right,mid+1,e,i,j);
    if(p1>p2)
        return p2;
    else
        return p1;
}


void update(int node,int b,int e,int i,int newvalu)
{
    if(i>e || i<b)
    {
        return ;
    }
    if(b>=i && e<=i)
    {
        tree[node]=newvalu;
        return;
    }
    int left=node*2;
    int right=(node*2)+1;
    int mid=(b+e)/2;
    update(left,b,mid,i,newvalu);
    update(right,mid+1,e,i,newvalu);
    if(tree[left]>tree[right])
    {
        tree[node]=tree[right];
    }
    else
        tree[node]= tree[left];
}
```

```
int main()
{
   int n,x,y,t,q;
   scanf("%d",&t);
   for(int i=1; i<=t; i++)
   {
      scanf("%d%d",&n,&q);
      for(int i=1; i<=n; i++)
      {
         scanf("%d",&arr[i]);
      }
      init(1,1,n);
      printf("Case %d:\n",i);
      while(q--)
      {
         scanf("%d%d",&x,&y);
         printf("%d\n",query(1, 1, n, x, y));
      }
   }
}
```

Find maximum value index i to index j in a given array:

```
#include<bits/stdc++.h>
#define mx 100009
using namespace std;
int tree[mx*3],arr[mx];

void init(int node,int b,int e)
{
   if(b==e)
   {
      tree[node]=arr[b];
      return;
   }
   int left=node*2;
   int right=(node*2)+1;
   int mid=(b+e)/2;
```

```
    init(left,b,mid);
    init(right,mid+1,e);
    if(tree[left]<tree[right])
    {
        tree[node]=tree[right];
    }
    else
        tree[node]= tree[left];
}

int query(int node,int b,int e,int i,int j)
{
    if(i>e || j<b)
    {
        return -100000000;
    }
    if(b>=i && e<=j)
    {
        return tree[node];
    }
    int left=node*2;
    int right=(node*2)+1;
    int mid=(b+e)/2;
    int p1=query(left,b,mid,i,j);
    int p2=query(right,mid+1,e,i,j);
    if(p1<p2)
        return p2;
    else
        return p1;
}


void update(int node,int b,int e,int i,int newvalu)
{
    if(i>e || i<b)
    {
        return ;
    }
    if(b>=i && e<=i)
    {
```

```
        tree[node]=newvalu;
        return;
    }
    int left=node*2;
    int right=(node*2)+1;
    int mid=(b+e)/2;
    update(left,b,mid,i,newvalu);
    update(right,mid+1,e,i,newvalu);
    if(tree[left]<tree[right])
    {
        tree[node]=tree[right];
    }
    else
        tree[node]= tree[left];
}


int main()
{
    int n,x,y,t,q;
    while(scanf("%d%d",&n,&q))
    {
        if(n==0)
            break;
        for(int i=1; i<=n; i++)
        {
            scanf("%d",&arr[i]);
        }
        init(1,1,n);
        while(q--)
        {
            scanf("%d%d",&x,&y);
            printf("%d\n",query(1, 1, n, x, y));
        }
    }
}
```

Segment tree Lazy propagation

Find sum of index i to index j in a given array and upade index i to j :

```
#include<bits/stdc++.h>
#define mx 100001
using namespace std;
int arr[mx];

struct info
{
  long long int prop,sum;
} tree[mx*3];

void init(int node,int b,int e)
{
   if(b==e)
   {
      tree[node].sum=arr[b];
      return;
   }
   int left=node*2;
   int right=(node*2)+1;
   int mid=(b+e)/2;
   init(left,b,mid);
   init(right,mid+1,e);
   tree[node].sum=tree[left].sum+tree[right].sum;
}

void update(int node,int b,int e,int i,int j,int x)
{
   if(i>e || j<b)
      return;
   if(b>=i && e<=j)
   {
      tree[node].sum+=((e-b+1)*x);
      tree[node].prop+=x;
      return;
   }
   int mid=(b+e)/2;
```

```
    int left=(node*2);
    int right=(node*2)+1;
    update(left,b,mid,i,j,x);
    update(right,mid+1,e,i,j,x);
    tree[node].sum=tree[left].sum+tree[right].sum+(e-b+1)*tree[node].prop;
}

int query(int node,int b,int e,int i,int j,int carry=0)
{
    if(i>e || j<b)
        return 0;
    if(b>=i && e<=j)
    {
        return tree[node].sum+carry*(e-b+1);
    }
    int mid=(b+e)/2;
    int left=(node*2);
    int right=(node*2)+1;
    int p1=query(left,b,mid,i,j,carry+tree[node].prop);
    int p2=query(right,mid+1,e,i,j,carry+tree[node].prop);
    return p1+p2;
}

int main()
{
    int n;
    cin>>n;
    for(int i=1; i<=n; i++)
    {
        cin>>arr[i];
    }
    init(1,1,n);
    cout << query(1, 1, n, 3, 5,0) << endl;
    update(1, 1, n, 3, 5,2);
    cout << query(1, 1, n, 3, 5,0) << endl;
}
```

STL Function

Pair :

Declare:

pair<int,int>p;

Example :

Pair<string,int>pr(“jon”,7);

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   pair<string,double>p("abc",3.24);
   pair<string,double>p2;
   pair<string,double>p3;
   p2.first="RANa";
   p2.second=0.99;
   p3=make_pair("rana",2.00);
   cout<<p.first<<" "<<p.second<<endl;
   cout<<p2.first<<" "<<p2.second<<endl;
   cout<<p3.first<<" "<<p3.second<<endl;
}
```

Pair Sort:

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

bool sortbysecond ( const pair<int,int>&a , const pair<int,int>&b ){
  return (a.second<b.second);
}
```

```
int main()
{
   int n, a, b;
   vector< pair<int, int> > A;
   scanf("%d", &n);
   for (int i = 0; i < n; i++)
   {
      scanf("%d %d", &a, &b),
           A.push_back(make_pair(a,b));
   }
   /// sort by first element
   sort(A.begin(), A.end());

   for (int i = 0; i < n; i++)
   {
      cout<<A[i].first<<" "<<A[i].second<<endl;
   }
    /// sort by second element
   sort(A.begin(), A.end(),sortbysecond);

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

Input :

```
 4
10 30
20 60
5  20
40 50

Sort by first:
5 20
```

10 30
20 60
40 50

Sort by second :
5 20
10 30
40 50
20 60

Set:

Basic operation:

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

set<string>m;
set<string>:: iterator it;

int main()
{
   int t;
   string s;
   cin>>t;
   while(t--)
   {
      cin>>s;
      m.insert(s);
   }
   for(it= m.begin(); it!=m.end(); it++)
   {
      cout<<*it<<endl;
   }
}
```

For element Finding in set:

```
set<int> s;
if(s.find(42) != s.end()) {
// 42 presents in set
}
 else {
 // 42 not presents in set
 }
```

Remove duplicate value of by set:

A simple way to get rid of duplicates in array:

```
int data[5] = { 5, 1, 4, 2, 3 };
set<int> S(data, data+5);
```

It gives us a simple way to get rid of duplicates in vector, and sort it:

```
 vector<int> v;
set<int> s(all(v));
 vector<int> v2(all(s));
```

Map :

```
#include<stdio.h>
using namespace std;

int main()
{
   map<char,int>Map;
```

```
    map<char,int>::iterator it;
    for(char c='a'; c<='j'; c++)
    {
        int value=(int)c;
        Map[c]=value;
    }

    cout<<"Map size: "<<Map.size()<<endl;
    cout<<"Map key value: \n";
    for(it=Map.begin(); it!=Map.end(); it++)
    {
        cout<<"Key-> "<<(*it).first<<" value-> "<<(*it).second<<endl;
/// OR :  cout<<"Key-> "<<it->first<<" value-> "<<it->second<<endl;
    }
    cout<<"Key a value: "<<Map['a']<<endl;

    if(Map.find('a')!=Map.end())cout<<"key 'a' found"<<endl;
    else cout<<"key 'a' not found"<<endl;

    if(Map.find('z')!=Map.end())cout<<"key 'z' found"<<endl;
    else cout<<"key 'z' not found"<<endl;
    Map.clear();
    cout<<"Is map empty: "<<Map.empty()<<endl;
}
```

Map with vector:

Problem: Uri 1244(sort by length. Given a line of string We need to sort word of this string by there length .If two word same length then which comes input first it's priority to place first)

Input:
1
Top coder comp wedn at midnight
Output:
Midnight coder comp wedn Top at

```
#include<bits/stdc++.h>
```

```
using namespace std;

int main()
{
   int ln,n;
   cin>>n;
   cin.ignore();
   while(n--)
   {
      map<int,vector<string> >p;
      string s , m;
      getline(cin , s);
      for(int i=0; i<s.size(); i++)
      {
         if(s[i]==' ')
         {
            ln=m.size();
            p[ln].push_back(m);
            m="";
         }
         else
         {
            m+=s[i];
         }
      }
      ln=m.size();
      p[ln].push_back(m);
      for(int j=60; j>=1; j--)
      {
         for(int i=0; i<p[j].size(); i++)
         {
            cout<<p[j][i]<<" ";
         }
      }
      cout<<endl;
   }
}
```

String:

Erase and insert and find function:

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

int main(){
string s;
getline(cin,s);
int index=s.find("ustc");
while(index!=-1){
   s.erase(index,4); /// index number and how many words need erase.
   s.insert(index,"iiuc");/// index number and which string need to insert.
   index=s.find("ustc");
}
cout<<s<<endl;
}
```

Input:

abcustcustc

output:

abciiuciiuc.

String Stream:

If problem set is not given any input which denote the how many input is taken.For this case we use Stringstream.We take the input as a string.

Example:

100 50 20 30 10 70

Code:

```
Char line[1000];

while( gets( line ) ) {

stringstream ss( line ); // initialize kortesi
int num;
```

```
vector< int > v;
while( ss >> num ) {
v.push_back( num );
} // :P
sort( v.begin(), v.end() );
}
```

Easy way to convert integer to string by Stringstream:

```
int a = 10;

stringstream ss;

ss << a;

string str = ss.str();
```

Stack :

Normal operation**:**

```
stack< int > st;
st.push( 10);  // inserting
st.push( 20);  // inserting
st.push(30) ;  // inserting

while( !st.empty() ) {
cout << st.top() << endl;   // printing the top
st.pop();  // removing that one
}
```

Uri code Parenthesis balance: 1068

```
int main()
{
  string b;
  while(getline(cin,b)){
      stack<char>s;
```

```
    for(int i=0;i<b.size();i++){
        if(s.size()>0 && s.top()=='(' && b[i]==')')
            s.pop();
        else if(b[i]=='(' || b[i]==')'){
            s.push(b[i]);
        }

    }
    if(s.size()>0){
        cout<<"incorrect"<<endl;
    }
    else{
        cout<<"correct"<<endl;
    }
   }
}
```

Queue:

Normal operation:

```
queue< int > q;
q.push( 10); // inserting
q.push(20) ; // inserting
q.push(30) ; // inserting

while( !q.empty() ) {
cout << q.front() << endl; // printing the front
q.pop(); // removing that one
}
```

Deque:

In the Double ended queue element add remove in the both front and back side.

deque declare:

```
deque< Type >Name;
deque< int >Name;
```

deque operation:
For element add and remove into the front use
push_front() and pop_back() .
For element add and remove into the back use
push_back() and pop_back() .

Priority Queue:
priority_queue< Type >Name;
priority_queue< int >Name;

In priority queue the front element have always lager element. By changing property we can put small element into the front in priority queue.

Priority queue operation:
priority queue operation same as queue operation. Only have top() instead front()

Longest Increasing subsequence in O(n^2) time:

```
#include <bits/stdc++.h>
using namespace std;
int a[10000],lis[10000];

int main()
{
int n;
   cin>>n;
   for(int i=0; i<n; i++)
   {
      cin>>a[i];
      lis[i]=1;
   }
   int ans=1;
   for(int j=1; j<n; j++)
   {
      for(int i=0; i<j; i++)
      {
         if(a[i]<=a[j] && lis[i]+1>lis[j])
         {
```

```
            lis[j]=lis[i]+1;
            ans=max(ans,lis[j]);
          }
       }
    }
    cout<<ans<<endl;
}
```

Longest Increasing subsequence in O(logn) time:

```
#include<bits/stdc++.h>
using namespace std;
int a[1000000],b[1000000],c[10000],ans[10000];

int main(){
vector<int>v;
int n,m,i=0;
int k=0;
while(cin>>m){
    b[i]=m;
    c[i]=m;
    vector<int>::iterator it=lower_bound(v.begin(),v.end(),m);
    if(v.end()==it){
        v.push_back(m);
        a[k]=i;       /// save index
        b[a[k]]=a[k-1]; /// assing parent of it's previous following parent
      ///  cout<<a[k]<<endl;
       /// cout<<b[a[k]]<<" val "<<a[k-1]<<endl;
        k++;
    }
    else{
        v[it-v.begin()]=m;
        a[it-v.begin()]=i;         /// save index
        b[a[it-v.begin()]]=a[(it-v.begin())-1];  /// assing parent of it's previous
following parent
       /// cout<<a[it-v.begin()]<<endl;
       /// cout<<b[a[it-v.begin()]]<<" val2 "<<a[(it-v.begin())-1]<<endl;
    }
    i++;
}
```

```
int temp=a[k-1],x=0;
for(int i=v.size()-1;i>=0;i--){
   ans[x++]=c[temp];
   temp=b[temp];
}
for(int i=x-1;i>=0;i--){
   cout<<ans[i]<<endl;
}
}
```

Matrix Expo:

```
#include<bits/stdc++.h>
using namespace std;
#define SZ 2
#define ll long long
ll M ;

struct Matrix{
   ll mat[SZ][SZ];
};
Matrix matMul(Matrix A,Matrix B)
{
   Matrix C;
   for(int i = 0 ; i < SZ ; i++)
      for(int j = 0 ;j < SZ ;j++)
      {
         C.mat[i][j] = 0 ;
         for(int k = 0 ; k < SZ ;k++){

C.mat[i][j]=((C.mat[i][j]%M)+(((A.mat[i][k]%M)*(B.mat[k][j]%M))%M))%M;
         }
      }
   return C;
}
Matrix matExpo(Matrix BASE,int p) // X^p
{
   if(p==1)return BASE;
   Matrix R = matExpo(BASE, p >> 1 ); // p>>1 means p/2.
   R = matMul(R,R);
   if(p&1) R = matMul(R,BASE); // check p is odd?
```

```
    return R;
}
int main()
{
    int test , cs = 1 ,n;
    Matrix Base = {
                    1 , 1,
                    1 , 0
                  };
    while(scanf("%d",&n)==1)
    {
      M =10000000+7 ;
      int ans = 0 ;
      if(n > 1 ){
           Matrix ret = matExpo(Base,n-1);
            ans =  ret.mat[0][0] ;
      }
      printf("%lld\n",ans);
    }
}
```

Binary Search Bisection Method:

Light oj problem:1138

You task is to find minimal natural number **N**, so that **N!** contains exactly **Q** zeroes on the trail in decimal notation. As you know **N! = 1*2*...*N**. For example, 5! = 120, 120 contains one zero on the trail.

Code:

```
#include <bits/stdc++.h>
using namespace std;
#define ft    int t; scanf("%d", &t); for(int c=1; c<=t; c++)
#define isc(a) scanf("%d", &a)
#define llsc(a) scanf("%lld", &a)
#define casepf(a) printf("Case %d: ", a)
typedef long long int ll;
#define sz 1000000002
```

```
int flag=0;
vector<int>p;

int f()
{
   int x=5,temp;
   while(x<=sz)
   {
      p.push_back(x);
      x=x*5;
   }
}

ll countt(ll m)
{
   ll x=0;
   for(int i=0; i<p.size(); i++)
   {
      x+=(ll)m/p[i];
   }
   return x;
}

ll bisection(ll n)
{
   ll low=1,high=sz,mid,res,temp=-1;
   while(low<=high)
   {
      mid=(low+high)/2;
      res=countt(mid);
      if(res==n)
      {
         temp=mid;
         high=mid-1;
      }
      else if(res>n)
      {
         high=mid-1;
      }
      else
```

```
        {
            low=mid+1;
        }
    }
    return temp;
}
int main()
{
    ll n,ans;
    f();
    ft
    {
        llsc(n);
        ans=bisection(n);
        casepf(c);
        if(ans==-1)
        {
            printf("impossible\n");
        }
        else{
            printf("%lld\n",ans);
        }
    }
}
```