1.

```
///sum of number of divisor between range 1 to n
#include<bits/stdc++.h>
using namespace std;
int Snod(int n)
{
   int sum=0;
   int u=sqrt(n);
   for(int i=1;i<=u;i++)
   {
      sum+=(n/i)-i;
   }
   sum*=2;
   sum+=u;
   return sum;
}
int main()
{
   int n;
   cout<<"N= ";
   cin>>n;
   cout<<"ans is = "<<Snod(n)<<endl;
   return 0;
}
```

2.

```
///big mode of power(input is a^b,c)
#include<bits/stdc++.h>
using namespace std;
int modrecursion(int a,int b,int c)
{
   if(b==0)
      return 1;
   if(b==1)
      return a%c;
   else if(b%2==0)
```

```
   {
      return modrecursion((a*a)%c,b/2,c);
   }
   else
    return (a*modrecursion((a*a%c),b/2,c))%c;
}
int main()
{
   int a,b,c;
   cout<<"enter a ,b,c \n";
   cin>>a>>b>>c;
   cout<<"ans is = "<<modrecursion(a,b,c)<<"\n";
}
```

3.

```
///simple binary search
#include<bits/stdc++.h>
using namespace std;
int save[20],n;
bool flag=0;
void binary()
{
   int number,mid;
   cout<<"what number do you wish to find?\n";
   cin>>number;
   int low,high;
   low=1;
   high=n;
   while(low<=high)
   {
      mid=(low+high)/2;
      if(save[mid]==number)
      {
         cout<<"found at "<<mid<<" index\n";
         flag=1;
         return;
```

```
        }
        if(save[mid]>number)
        {
            high=mid-1;
        }
        else if(save[mid]<number)
        {
            low=mid+1;
        }
    }
}
int main()
{
    cout<<"How many numbers?\n";
    cin>>n;
    for(int i=1; i<=n; i++)
    {
        cin>>save[i];
    }
    sort(save+1,save+n);
    for(int i=1; i<=n; i++)
    {
        cout<<save[i]<<" ";
    }
    cout<<endl;
    flag=0;
    binary();
    if(!flag)
    {
        cout<<"element not found!\n";
    }
    return 0;
}
```

4.

```
///counting the number of divisor
```

```
/// it will only give the total number of divisor,to get the divisor(use loop method);
#include<bits/stdc++.h>
using namespace std;
bitset<10000010> bs;
typedef long long int ll;
typedef vector<int>vi;
vi primes;
vi divisor;
ll seive_size;
void seive(ll upperbound)
{
   seive_size=upperbound+2;
   bs.set(); /// shobgular value 1 kore dilam
   bs[0]=bs[1]=0;
   primes.push_back(2);
   for ( int i = 4; i <= seive_size; i += 2 )
   {
      bs[i] = 0;
   }
   ll sqrtn =sqrt( seive_size );
   for(ll i=3; i<=sqrtn; i=i+2) /// we dont want even number to check
   {
      if(bs[i])
      {
         for(ll j=i*i; j<=seive_size; j=j+(2*i)) /// omitting even,, 9,15,21.....
         {
            bs[j]=0;
         }
         primes.push_back((int)i);
      }
   }
}
int nod (ll n)
{
   int sqrtn=sqrt(n);
```

```
    int ans=1;
    for(int i=0; i<primes.size()&&primes[i]<=sqrtn; i++)
    {
        if(bs[n]) /// if n is a prime,,then it cant be reduced anymore
            break;
        if(n%primes[i]==0)
        {
            int power=0;
            while(n%primes[i]==0)
            {
                n/=primes[i];
                power++;
            }
            sqrtn=sqrt(n);
            ans=ans*(power+1);
        }
    }
    if(n!=1)
    {
       ans=ans*2;
    }
    return ans;
}
int main()
{
    seive(10000000);
    cout<<"enter the number to find its factorization\n";
    int n;
    cin>>n;
    int r=nod(n);
    cout<<n<<" has total "<<r<<" divisors\n";
}
```

5.

```
///counting the divisor,loop_method
#include<bits/stdc++.h>
```

```
using namespace std;
typedef long long int ll;
typedef vector<int> vi;
vector<int>divisor;
int nod(int n)
{
   int counter=0;
   int sqrtn=sqrt(n);
   for(int i=1; i<=sqrtn; i++)
   {
      if(n%i==0)
      {
         int take=i;
         int take2=n/i;
         if(take!=take2)
         {
            divisor.push_back(take);
            divisor.push_back(take2);
            counter=counter+2;
         }
         else
         {
            counter++;
            divisor.push_back(take);
         }
      }
   }
   return counter;
}
int main()
{
   int n;
   cout<<"enter the number to find its total divisor\n";
   cin>>n;
   cout<<"ans is "<<nod(n)<<endl;
```

```
    cout<<"and the divisors are\n";
    for(int i=0; i<divisor.size(); i++)
    {
        cout<<divisor[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

6.

```
///decimal to any base,,even 16,who cares
#include<bits/stdc++.h>
using namespace std;
char symbol[] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
string convert(int x,int base)
{
    string str="";
    while(x!=0)
    {
        int r=x%base;
        str=str+symbol[r];
        x=x/base;
    }
    if(str.size()==0)
    {
        str=symbol[0];
    }
    reverse(str.begin(),str.end());
    return str;
}
int main()
{
    int number;
    while(1)
    {
        cout<<"enter a number\n";
```

```
    int nunber;
    cin>>number;
    cout<<"enter the base you wish to convert\n";
    int base;
    cin>>base;
    string str;
    str=convert(number,base);
    cout<<"Ans is "<<str<<endl;
  }
  return 0;
}
```

7.

```
///count the number of positive integers<n that are relatively prime to n
#include<bits/stdc++.h>
using namespace std;
bitset<10000010> bs;
typedef long long int ll;
typedef vector<int>vi;
vi primes;
ll seive_size;
void seive(ll upperbound)
{
  seive_size=upperbound+2;
  bs.set(); /// shobgular value 1 kore dilam
  bs[0]=bs[1]=0;
  primes.push_back(2);
  for ( int i = 4; i <= seive_size; i += 2 )
  {
    bs[i] = 0;
  }
  ll sqrtn =sqrt( seive_size );
  for(ll i=3; i<=sqrtn; i=i+2) /// we dont want even number to check
  {
    if(bs[i])
    {
```

```
            for(ll j=i*i; j<=seive_size; j=j+(2*i)) /// omitting even,, 9,15,21.....
            {
               bs[j]=0;
            }
            primes.push_back((int)i);
        }
    }
}
ll primefactor(ll n)
{
    ll pf_idx=0,pf,ans=n;
    pf=primes[pf_idx];
    while(pf*pf<=n)
    {
        if(n%pf==0)
        {
          ans-=ans/pf;
        }
        while(n%pf==0)
        {
            n/=pf;

        }
        pf=primes[++pf_idx];
    }
    if(n!=1)
    {
        ans-=ans/n;
    }
    return ans;
}
int main()
{
    seive(10000000);
    cout<<"enter the number\n";
```

```
   int n;
   cin>>n;
   ll r=primefactor(n);
   cout<<"Total number is= "<<n<<" is = "<<r<<endl;
}
```

8.

```
///solve_linear_diopantine_equation with ext_gcd()
#include<bits/stdc++.h>
using namespace std;
int d,x,y,g;
int gcd(int a,int b)
{
   return b==0?a:gcd(b,a%b);
}
void ex_gcd(int a,int b)
{
   if(b==0)
   {
      d=a;
      x=1;
      y=0;
   }
   else
   {
      ex_gcd(b,a%b);
      int temp=x;
      x=y;
      y=temp-(a/b)*y;
   }
}
bool linear_diop(int A,int B,int C)
{
    g=gcd(A,B);
    cout<<"g= "<<g<<endl;
    if(C%g!=0)
```

```
    {
      return false ;
    }
    int a,b,c;
    a=A/g;b=B/g;c=C/g;
    ex_gcd(a,b);
    if ( g < 0 ) { //Make Sure gcd(a,b) = 1
       a *= -1; b *= -1; c *= -1;
    }
    x=x*c;
    y=y*c;
    return true;
}
int main()
{
    int a,b,c;
    cout<<"Enter the value of A,B,C\n";
    cin>>a>>b>>c;
    bool check;
    check=linear_diop(a,b,c);
    if(!check)
       cout<<"NO solution is possible\n";

    else
    {
      cout<<"possible solution is  "<<x<<" "<<y<<endl;

       int k = 1; //Use different value of k to get different solutions
       printf ( "Another Possible Solution (%d %d)\n", x + k * ( b / g ), y - k * ( a / g ) );
    }
    return 0;
}
```

9.

```
///best for factorial factorization
#include<bits/stdc++.h>
```

```
using namespace std;
bitset<10000010> bs;
typedef long long int ll;
typedef vector<int>vi;
map<ll,ll>prime_list;
vi primes;
ll seive_size;
void seive(ll upperbound)
{
   seive_size=upperbound;
   bs.set(); /// shobgular value 1 kore dilam
   bs[0]=bs[1]=0;
   primes.push_back(2);
   for ( int i = 4; i <= seive_size; i += 2 )
   {
      bs[i] = 0;
   }
   for(ll i=3; i<=seive_size; i=i+2) /// we dont want even number to check
   {
      if(bs[i])
      {
         for(ll j=i*i; j<=seive_size; j=j+(2*i)) /// omitting even,, 9,15,21.....
         {
            bs[j]=0;
         }
         primes.push_back((int)i);
      }
   }
}
void  factorial_facto ( int n )
 {

   for(int i=0; i<primes.size()&&primes[i]<=n;i++)
   {
      int x = n;
```

```
        int freq = 0;
        while ( x / primes[i] ) {
            freq += x / primes[i];
            x = x / primes[i];
        }
        prime_list[primes[i]]=freq;
    }
}
int main()
{
    cout<<"enter the number to find its factorial_factorization\n";
    ll n;
    cin>>n;
    seive(n);
    factorial_facto(n);
    for(int i=0;i<primes.size();i++)
    {
        printf ("%d^%d\n",primes[i],prime_list[ primes[i]]);
    }
    cout<<endl;
}
```

10.

```
///general subset using backtrack
#include<bits/stdc++.h>
using namespace std;
int n,save[20];
vector<int>v;
void backtrack(int checker)
{
    if(v.size()==n)
    {
        for(int i=0; i<n; i++)
        {
            cout<<v[i]<<" ";
        }
```

```
        return;
    }
    for(int i=checker; i<n; i++)
    {
        cout<<"number= "<<save[i]<<endl;
        v.push_back(save[i]);
        backtrack(i+1);
        v.pop_back();
    }
}
int main()
{
    cout<<"How many number\n";
    cin>>n;
    for(int i=0; i<n; i++)
    {
        cin>>save[i];
    }
    backtrack(0);
    return 0;
}
```

14.

```
///large factorial
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a[500],size=1;
    int n;
    cin>>n;
    a[0]=1;
    int carry=0;
    for(int i=n;i>=2;i--)
    {
          for(int j=0;j<size;j++)
```

```
        {
           carry=a[j]*i+carry;
           a[j]=carry%10;
           carry=carry/10;
        }
        while(carry>0)
        {
         a[size]=carry%10;
         size++;
         carry=carry/10;
        }
   }
   for(int i=size-1;i>=0;i--)
   {
      cout<<a[i];
   }
   return 0;
}
```

15.

```
///max power of 2 between a range
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int number;
   cout<<"Enter a number\n";
   cin>>number;
   number=number|(number>>1);
   number=number|(number>>2);
   number=number|(number>>4);
   number=number|(number>>8);
   number=number|(number>>16);
   cout<<((number+1)>>1);
   return 0;
}
```

16.

```
///N_queen_column_based
#include<bits/stdc++.h>
using namespace std;
bool flag=false;
int counter=0;
int save2[100],save[100],minn;
bool place(int r,int c)
{
   for(int col=1;col<c;col++)
   {
     int row=save2[col];
     if(r==row)
     {
        return 0;
     }
     if(abs(col-c)==abs(row-r))
     {
        return 0;
     }
   }
   return true;
}
void backtrack(int c)
{
 if(c>8)
 {
     for(int i=1;i<=8;i++)
     {
       // cout<<"save[i]= "<<save[i]<<" save2[i]= "<<save2[i]<<endl;
        cout<<"row= "<<save[i]<<"\tand column= "<<i<<endl;
     }
     //cout<<counter<<endl;
     minn=min(minn,counter);
     counter=0;
```

```
    return;
 }
 else
 {
    for(int r=1;r<=8;r++)
    {
     if(place(r,c))
     {
        save2[c]=r;
        backtrack(c+1);
        save2[c]=0;
     }
    }
 }
}
int counter2=1;
int main()
{
   backtrack(1);
   return(0);
}
```

17.

```
///Nqueen row based
///i changed the code from cp3,,they mixed up row and column variable,,it was getting difficult for me
#include<bits/stdc++.h>
using namespace std;
int save[20],n;
bool flag=false;
bool place(int r,int c)
{
   int column;
   for(int row=1; row<r; row++)
   {
      column=save[row];
      /// here i is the row and column=save[row];
```

```
      //cout<<"column= "<<column<<"and c= "<<c<<endl;
      //cout<<"abs(row-r)= "<<abs(row-r)<<" and abs(column-c)= "<<abs(column-c)<<endl;
      if(column==c)
      {
         return false;
      }
      if(abs(row-r)==abs(column-c))
      {
         return false;
      }
   }
   return true;
}
void hold(int r)
{
   //cout<<"value of r is= "<<r<<endl;
   if(r>n)
   {
      cout<<"you can place the queen in following order\n";
      for(int i=1;i<=n;i++)
      {
         cout<<"row= "<<i<<" \tcolumn= "<<save[i]<<endl;
      }
      return;
   }
   else
   {
      for(int i=1; i<=n; i++)
      {
         if(place(r,i))
         {
            int take=r;
            save[r]=i;
            hold(++take);  //to use a temp variable for r important or hold(r+1) will be good.
            save[r]=0  /// Unless you are not planning to printe the whole array,,ei line dorkar nai
```

```
        }
      }
   }
}
int main()
{
   memset(save,0,sizeof(save));
   cout<<"how many queen?\n";
   cin>>n;
   hold(1);
   cout<<"Bazinga!\n";
   return(0);
}
```

18.

```
///power of 2 check with bit manipulation
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int number;
   cout<<"Enter a number\n";
   cin>>number;
   if(number&&!(number&(number-1)))     /// because x and x-1 er & hobee always 0
   {
      cout<<"It is a power of 2 man!\n";
   }
   else
   {
      cout<<"It is not\n";
   }
   return 0;
}
```

19.

```
///prime_facto with trail division method improved
#include<bits/stdc++.h>
```

```
using namespace std;
bitset<10000010> bs;
typedef long long int ll;
typedef vector<int>vi;
vi primes;
//vi primefactor;
ll seive_size;
void seive(ll upperbound)
{
   seive_size=upperbound+2;
   bs.set(); /// shobgular value 1 kore dilam
   bs[0]=bs[1]=0;
   primes.push_back(2);
   for ( int i = 4; i <= seive_size; i += 2 )
   {
      bs[i] = 0;
   }
   ll sqrtn =sqrt( seive_size );

   for(ll i=3; i<=sqrtn; i=i+2) /// we dont want even number to check
   {
      if(bs[i])
      {
         for(ll j=i*i; j<=seive_size; j=j+(2*i)) /// omitting even,, 9,15,21.....
         {
            bs[j]=0;
         }
         primes.push_back((int)i);
      }
   }
}

vi primefactor(ll n)
{
```

```
    vi factor;
    int sqrtn=sqrt(n);
//    while(pf*pf<=n)
//    {
//        while(n%pf==0)
//        {
//            n/=pf;
//            factor.push_back((int)pf);
//        }
//        cout<<"pf= "<<pf<<" and n = "<<n<<endl;
//        pf=primes[++pf_idx];
//    }
    for(int i=0; i<primes.size()&&primes[i]<=sqrtn;i++)
    {
        if(bs[n]) /// if n is a prime,,then it cant be reduced anymore
            break;

        if(n%primes[i]==0)
        {
            while(n%primes[i]==0)
            {
                n/=primes[i];
                factor.push_back((int)primes[i]);
            }
            sqrtn=sqrt(n);
        }
    }
    if(n!=1)
    {
        factor.push_back((int)n);
    }
    return factor;
}

int main()
```

```
{
   seive(10000000);
   cout<<"checking seive\n";
   cout<<"enter -1 to break\n";
   int number;
   while(1)
   {
      cin>>number;
      if(number==-1)
         break;
      if(bs[number])
         cout<<"prime!\n";
      else
         cout<<"not prime!\n";
   }
   cout<<"enter the number to find its factorization\n";
   int n;
   cin>>n;
   vi r=primefactor(n);

   for(vi::iterator i=r.begin(); i!=r.end(); i++)
   {
      cout<<*i<<" ";
   }
   cout<<endl;
}
```

20.

```
///segmented seive
#include<bits/stdc++.h>
using namespace std;
typedef long long int ll;
typedef vector<int> vi;
vector<int>primes;
bitset<10000000> bs;
int arr[10000000];
```

```
ll seive_size;
void seive(ll upperbound)
{
   seive_size=upperbound+2;
   bs.set(); /// shobgular value 1 kore dilam
   bs[0]=bs[1]=0;
   primes.push_back(2);
    for ( int i = 4; i <= seive_size; i += 2 )
    {
        bs[i] = 0;
    }
   ll sqrtn =sqrt( seive_size );
   for(ll i=3;i<=seive_size;i=i+2) /// we dont want even number to check
   {
      if(bs[i])
      {
       for(ll j=i*i;j<=seive_size;j=j+(2*i))  /// omitting even,, 9,15,21.....
       {
          bs[j]=0;
       }
       primes.push_back((int)i);
      }
   }
}
void segmented_seive(ll a,ll b)
{

   int sizee=sqrt(b);
   seive(sizee);
   memset (arr,0,sizeof arr );
   if(a==1)
      a++;
   for(int i=0;i<primes.size()&&primes[i]<=sizee;i++)
   {
      int p=primes[i];
```

```
        int j=p*p;
        if(j<a)
        {
           j=ceil(a/(double)p)*p;
        }
        for(;j<=b;j+=p)
        {
           arr[j-a]=1;
        }
     }
}
int main()
{
    ll a,b;
    cin>>a>>b;
    segmented_seive(a,b);
    cout<<"-1 to break \n";
    while(1)
    {
        cout<<"enter a number\n";
        int number;
        cin>>number;
        if(number<0)
           break;
        if(!arr[number-a])
        {
           cout<<"it is a prime!\n";
        }
        else
           cout<<"not prime\n";
    }
    return 0;
}
```

21.

```
///seive method the best one
```

```
#include<bits/stdc++.h>
using namespace std;
typedef long long int  ll;
bitset<10000010> bs;
vector<int>primes;
ll seive_size=0;
void seive(ll upperbound)
{
   seive_size=upperbound+2;
   bs.set(); /// shobgular value 1 kore dilam
   bs[0]=bs[1]=0;
   primes.push_back(2);
    for ( int i = 4; i <= seive_size; i += 2 )
    {
        bs[i] = 0;
    }
   ll sqrtn =sqrt( seive_size );
   for(ll i=3;i<=sqrtn;i=i+2) /// we dont want even number to check
   {
      if(bs[i])
      {
       for(ll j=i*i;j<=seive_size;j=j+(2*i))  /// omitting even,, 9,15,21.....
       {
          bs[j]=0;
       }
       primes.push_back((int)i);
      }
   }
}
bool isprime(int n)
{
   if(n<=seive_size)
      return bs[n];
   /// otherwise in terms of large prime
   for(int i=0;i<primes.size();i++)
```

```
   {
      if(n%primes[i]==0)
         return false ;
   }
   return true;
}
int main()
{
  seive(10000000);
  int n;
  cout<<"press 0 to terminate the process\n";
  while(1)
  {
     cin>>n;
     if(n==0)
     {
        break;
     }
     bool tag=isprime(n);
     if(tag)
     {
        cout<<n<<" is a prime\n";
     }
     else
       cout<<"the number is not a prime\n";
  }
}
```

22.

```
///sum of divisor of N loop method
#include<bits/stdc++.h>
using namespace std;
typedef long long int ll;
typedef vector<int> vi;
vector<int>divisor;
int sumnod(int n)
```

```
{
   int sum=0;
   int sqrtn=sqrt(n);
   for(int i=1; i<=sqrtn; i++)
   {
      if(n%i==0)
      {
         int take=i;
         int take2=n/i;

         if(take!=take2)
         {
            divisor.push_back(take);
            divisor.push_back(take2);
            sum+=take+take2;
         }
         else
         {
            divisor.push_back(take);
            sum+=take;
         }
      }
   }
   return sum;
}
int main()
{
   int n;
   cout<<"enter the number to find its total divisor\n";
   cin>>n;
   cout<<"Sum of the divisor is "<<sumnod(n)<<endl;
   cout<<"and the divisors are\n";
   for(int i=0; i<divisor.size(); i++)
   {
      cout<<divisor[i]<<" ";
```

```
    }
    cout<<endl;

    return 0;
}
```

23.

```
///sum of divisor of N prime facto method
#include<bits/stdc++.h>
using namespace std;
bitset<10000010> bs;
typedef long long int ll;
typedef vector<int>vi;
vi primes;
vi divisor;
ll seive_size;
void seive(ll upperbound)
{
    seive_size=upperbound+2;
    bs.set(); /// shobgular value 1 kore dilam
    bs[0]=bs[1]=0;
    primes.push_back(2);
    for ( int i = 4; i <= seive_size; i += 2 )
    {
        bs[i] = 0;
    }
    ll sqrtn =sqrt( seive_size );
    for(ll i=3; i<=sqrtn; i=i+2) /// we dont want even number to check
    {
        if(bs[i])
        {
            for(ll j=i*i; j<=seive_size; j=j+(2*i)) /// omitting even,, 9,15,21.....
            {
                bs[j]=0;
            }
            primes.push_back((int)i);
```

```
        }
    }
}
int nod (ll n)
{
    int res = 1;
    int sqrtn = sqrt ( n );
    for ( int i = 0; i < primes.size() && primes[i] <= sqrtn; i++ )
    {
        if ( n % primes[i] == 0 )
        {
            int tempSum = 1; //Contains value of (p^0+p^1+...p^a)
            int p = 1;

            while ( n % primes[i] == 0 )
            {
                n /= primes[i];
                p *= primes[i]; /// powering up
                tempSum += p;
            }
            sqrtn = sqrt ( n );
            res *= tempSum;
        }
    }
    if ( n != 1 )
    {
        res *= ( n + 1 ); //Need to multiply (p^0+p^1)
    }
    return res;
}
int main()
{
    seive(10000000);

    cout<<"enter the number to find its sum of divisor\n";
```

```
   int n;
   cin>>n;
   cout<<"Ans is = "<<nod(n)<<endl;
}
```

24.

```
///sum of number of divisor between range 1 to n
#include<bits/stdc++.h>
using namespace std;
int Snod(int n)
{
   int sum=0;
   int u=sqrt(n);
   for(int i=1;i<=u;i++)
   {
      sum+=(n/i)-i;
   }
   sum*=2;
   sum+=u;
   return sum;
}
int main()
{
   int n;
   cout<<"N= ";
   cin>>n;
   cout<<"ans is = "<<Snod(n)<<endl;
   return 0;
}
```