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;
}
```

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);

}
```

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;

}

//primefactor function


vi primefactor(ll n)

{

    vi factor;

    int sqrtn=sqrt(n);

    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 ;     }
```