1. Light More Light

Program-

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class Main {

	//bisection method :/ Math.sqrt is inaccurate.
	//speed = O(log2 n)
	public static long squareRoot (long l) {
		long min=0;
		long max=((long)Integer.MAX_VALUE)*2;
		long mid=0;
		while (min<=max) {
			mid=(min+max)/2;
			long value=mid*mid;
			if (value==l) {
				break;
			} else if (value<l) {
				min=mid+1;
			} else {
				max=mid-1;
			}
		}
		return mid;
	}

	public static void main(String[] args)  throws IOException {
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
		String s;
		while ((s=br.readLine())!=null) {
			long l=Long.parseLong(s);
			if (l==0) {
				break;
			}
			long sqrt=squareRoot(l);
			if (sqrt*sqrt==l) {
				System.out.println("yes");
			} else {
				System.out.println("no");
			}
		}
	}
}
```

Output-

```
input
3
no
6241
yes
8191
no
0


...Program finished with exit code 0
Press ENTER to exit console.
```

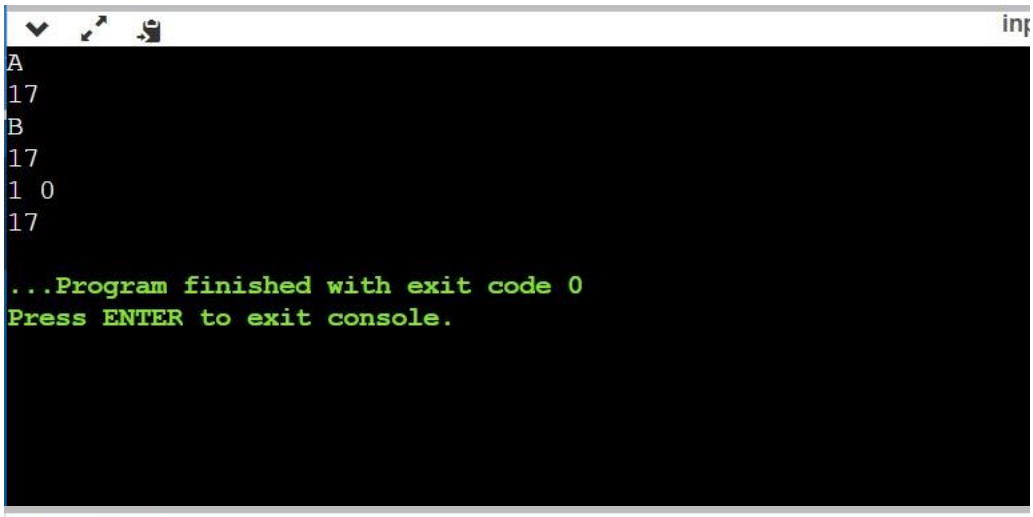
2. Euclid Problem

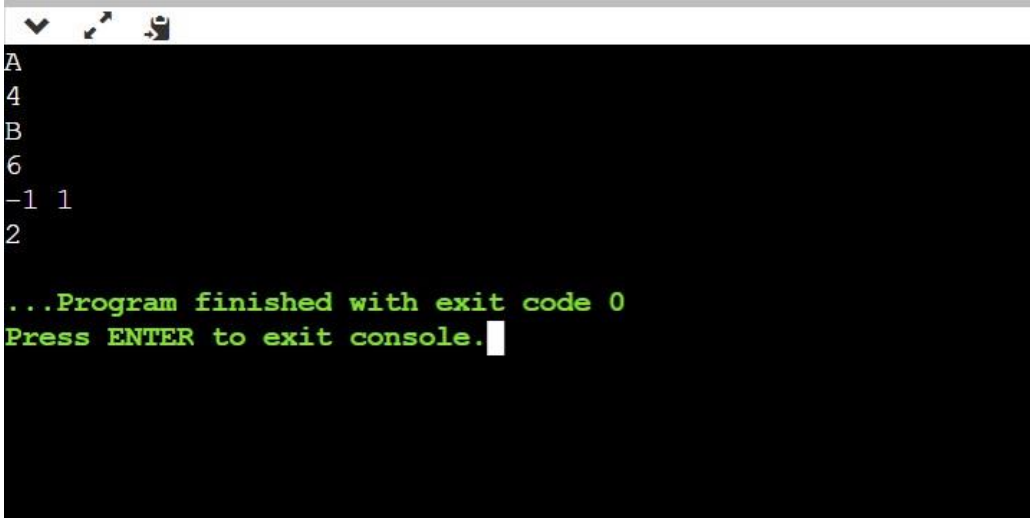
Program-

```
import java.util.*;

public class Main {
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      System.out.println("A");
      int A=sc.nextInt();
      System.out.println("B");
      int B=sc.nextInt();
      int X,Y;
      int i,D=1;

     for( i = 1; i <= A && i <= B; ++i)
      {
      if (A % i ==0 && B % i == 0) {
         D = i;
      }
       }

    for(X=A;X>=-1;--X){
       for(Y=B;Y>=0;--Y){
```

```
        if(A==B){
            if(A*X+B*Y==D && X>0){
             System.out.println(X+" " +Y);
             break;
         }
        }
        else{
         if(A*X+B*Y==D){
             System.out.println(X+" " +Y);
             break;
         }
        }

    }
    }
     //System.out.println("Hello World!");
    System.out.println("now: " + D);
   }
}
```

Output-





3. Factovisors

Program-

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter a no for n");
      int n=sc.nextInt();
      System.out.println("Enter a no for m");
      int m=sc.nextInt();
      int fact=1;
      for(int i=1;i<=n;i++){
        fact=fact*i;
      }
      if(fact%m==0){
        System.out.println(m+" divides "+n +"!");
      }
      else{
          System.out.println(m+" does not divides "+n+"!");
      }
      System.out.println("now: " +fact);
   }
}
```

Output-

```
Enter a no for n
100000
Enter a no for m
20
20 divides 100000!


...Program finished with exit code 0
Press ENTER to exit console.
```

```
Enter a no for n
9
Enter a no for m
6
6 divides 9!


...Program finished with exit code 0
Press ENTER to exit console.
```

4. Summation of Four Primes

Program-

```
import java.util.*;

public class Main {
  static int a = 0, b = 0;
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
            int n =sc.nextInt();
            generate(n);


    }

  static int isPrime(int x)
            {
                        // int s = (int)Math.sqrt(x);

       boolean y=true;
       if(x==1 || x==0){
             y=false;

       }
       for(int i=2;i<x/2;i++){
         if(x%i==0){
            y=false;
            break;
         }

       }
       if(y){
            // System.out.println("Prime");
            return 1;
       }
       else{
          return 0;
       }
            }

            static void Num(int x)
            {
                        // iterates to check prime
                        // or not
                        for (int i = 2; i <= x / 2; i++) {

                                    // calls function to check
                                    // if i and x-i is prime
                                    // or not
                                    if (isPrime(i) != 0 && isPrime(x - i) != 0) {
```

```
                    a = i;
                    b = x - i;

                    // if two prime numbers
                    // are found, then return

                }
            }
        }

        static void generate(int n)
        {
            if (n <= 7)
                System.out.println("Impossible");
            // if it is not even then 2 and 3
            // are first two of sequence
            if (n % 2 != 0) {// NOt Even
                Num(n - 5);
                System.out.println("2 3 " + a + " " + b);
            }
            else {// Even
                Num(n - 4);
                System.out.println("2 2 " + a + " " + b);
            }
        }

}
```

Output-

```
13
2 3 4 4


...Program finished with exit code 0
Press ENTER to exit console.
```

```
50
2 2 23 23


...Program finished with exit code 0
Press ENTER to exit console.
```

1.15-Puzzle Problem

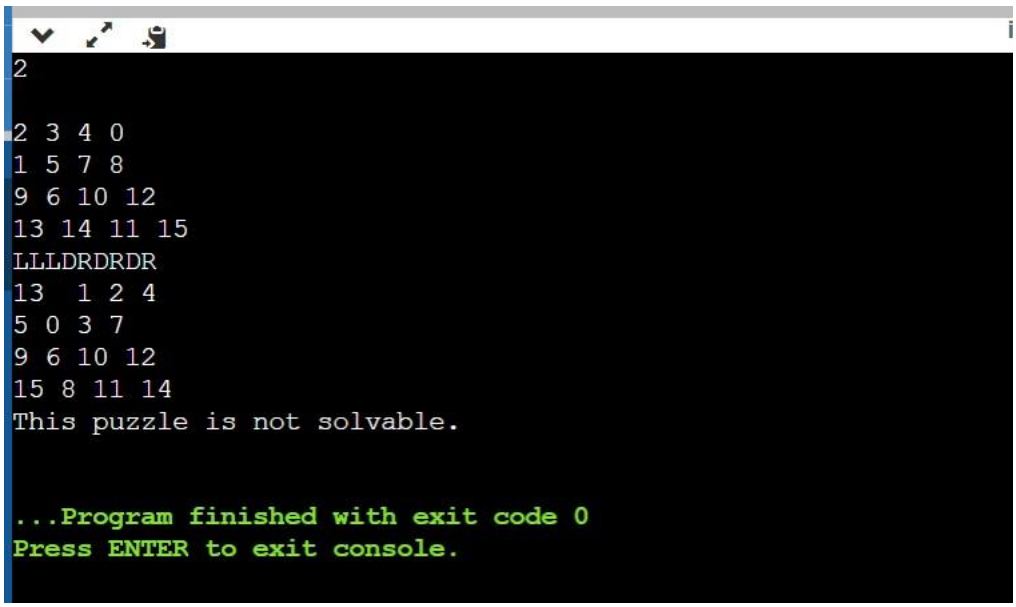
Program-

```
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
#define LLU unsigned long long
using namespace std;
struct status {
   char board[4][4];
   int ix, iy;
} init;
int pos[16][2], mxdep;
int dir[4][2] = {{0,-1},{-1,0},{1,0},{0,1}};/*u,l,r,d*/
char dirc[4] = {'L', 'U', 'D', 'R'}, path[100];
int solved;
bool solvable() {
   int sum = 0, row, i, j;
   for(i = 0; i < 16; i++) {
      if(init.board[i/4][i%4] == 0) {
         row = i/4 + 1;
         continue;
      }
      for(j = i+1; j < 16; j++) {
         if(init.board[j/4][j%4] < init.board[i/4][i%4]) {
            if(init.board[j/4][j%4])
               sum++;
         }
      }
   }
   return 1-(sum+row)%2;
}
int H() {
   static int i, j, sum, num;
   sum = 0;
   for(i = 0; i < 4; i++) {
      for(j = 0; j < 4; j++) {
         num = init.board[i][j];
         if(num == 0)
            continue;
         sum += abs(i-pos[num][0]) + abs(j-pos[num][1]);
      }
   }
   return sum;
}
int Htable[4][4][16];
int IDA(int dep, int hv, int prestep) {
   if(hv == 0) {
      solved = dep;
      path[dep] = '\0';
```

```
        puts(path);
        return dep;
    }
    if(dep + 5*hv/3 > mxdep) {
        return dep + 5*hv/3;
    }
    int i, tx, ty, x = init.ix, y = init.iy;
    int submxdep = 0xfff, val = 0xfff, shv;

    for(i = 0; i < 4; i++) {
        if(i + prestep == 3)    continue;
        tx = x + dir[i][0], ty = y + dir[i][1];
        if(tx < 0 || ty < 0 || tx > 3 || ty > 3)
            continue;

        shv = hv;
        shv -= Htable[tx][ty][init.board[tx][ty]];
        shv += Htable[x][y][init.board[tx][ty]];
        init.ix = tx, init.iy = ty;
        swap(init.board[x][y], init.board[tx][ty]);

        path[dep] = dirc[i];
        val = IDA(dep+1, shv, i);

        swap(init.board[x][y], init.board[tx][ty]);
        init.ix = x, init.iy = y;
        if(solved)  return solved;
        submxdep = min(submxdep, val);
    }
    return submxdep;
}
int main() {
    int test, i, j, k, initH;
    int cases = 0;
    for(i = 0, k = 0; i < 4; i++)
        for(j = 0; j < 4; j++)
            pos[++k][0] = i, pos[k][1] = j;
    for(i = 0; i < 4; i++)
        for(j = 0; j < 4; j++)
            for(k = 1; k < 16; k++)
                Htable[i][j][k] = abs(i - pos[k][0]) + abs(j - pos[k][1]);
    scanf("%d", &test);
    while(test--) {
        cases++;
        for(i = 0; i < 4; i++) {
            for(j = 0; j < 4; j++) {
                scanf("%d", &k);
                init.board[i][j] = k;
                if(init.board[i][j] == 0) {
                    init.ix = i, init.iy = j;
                }
```

```
        }
      }
      if(solvable()) {
        solved = 0, initH = mxdep = H();
        if(!mxdep) {
          puts("");
          continue;
        }
        while(solved == 0)
          mxdep = IDA(0, initH, -1);
        //printf("%d\n", solved);
      }else {
        puts("This puzzle is not solvable.");
      }
    }
    return 0;
}
```

Output-



2.Tug of War

Program-

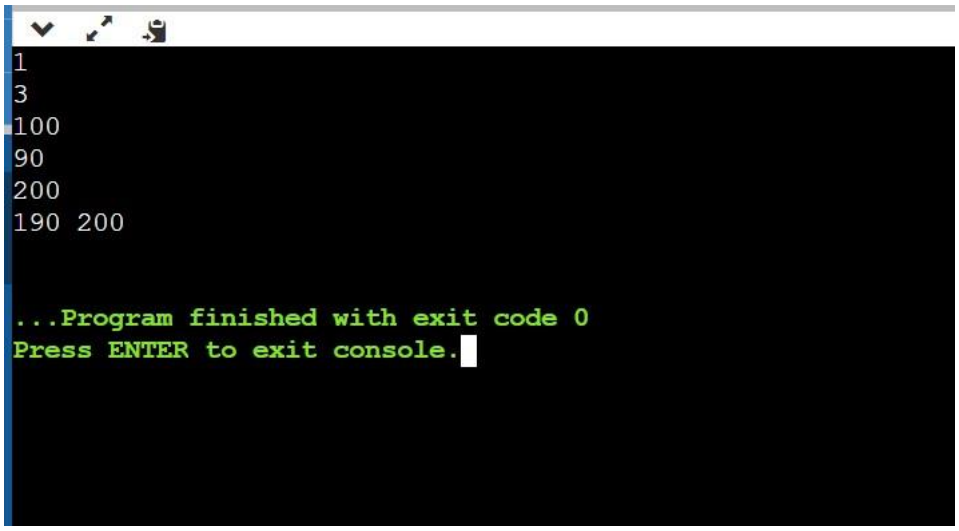
```
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
#define LLU unsigned long long
using namespace std;
struct status {
    char board[4][4];
    int ix, iy;
} init;
```

```
int pos[16][2], mxdep;
int dir[4][2] = {{0,-1},{-1,0},{1,0},{0,1}};/*u,l,r,d*/
char dirc[4] = {'L', 'U', 'D', 'R'}, path[100];
int solved;
bool solvable() {
   int sum = 0, row, i, j;
   for(i = 0; i < 16; i++) {
      if(init.board[i/4][i%4] == 0) {
         row = i/4 + 1;
         continue;
      }
      for(j = i+1; j < 16; j++) {
         if(init.board[j/4][j%4] < init.board[i/4][i%4]) {
            if(init.board[j/4][j%4])
               sum++;
         }
      }
   }
   return 1-(sum+row)%2;
}
int H() {
   static int i, j, sum, num;
   sum = 0;
   for(i = 0; i < 4; i++) {
      for(j = 0; j < 4; j++) {
         num = init.board[i][j];
         if(num == 0)
            continue;
         sum += abs(i-pos[num][0]) + abs(j-pos[num][1]);
      }
   }
   return sum;
}
int Htable[4][4][16];
int IDA(int dep, int hv, int prestep) {
   if(hv == 0) {
      solved = dep;
      path[dep] = '\0';
      puts(path);
      return dep;
   }
   if(dep + 5*hv/3 > mxdep) {
      return dep + 5*hv/3;
   }
   int i, tx, ty, x = init.ix, y = init.iy;
   int submxdep = 0xfff, val = 0xfff, shv;

   for(i = 0; i < 4; i++) {
      if(i + prestep == 3)    continue;
      tx = x + dir[i][0], ty = y + dir[i][1];
      if(tx < 0 || ty < 0 || tx > 3 || ty > 3)
```

```
            continue;

        shv = hv;
        shv -= Htable[tx][ty][init.board[tx][ty]];
        shv += Htable[x][y][init.board[tx][ty]];
        init.ix = tx, init.iy = ty;
        swap(init.board[x][y], init.board[tx][ty]);

        path[dep] = dirc[i];
        val = IDA(dep+1, shv, i);

        swap(init.board[x][y], init.board[tx][ty]);
        init.ix = x, init.iy = y;
        if(solved)  return solved;
        submxdep = min(submxdep, val);
    }
    return submxdep;
}
int main() {
    int test, i, j, k, initH;
    int cases = 0;
    for(i = 0, k = 0; i < 4; i++)
        for(j = 0; j < 4; j++)
            pos[++k][0] = i, pos[k][1] = j;
    for(i = 0; i < 4; i++)
        for(j = 0; j < 4; j++)
            for(k = 1; k < 16; k++)
                Htable[i][j][k] = abs(i - pos[k][0]) + abs(j - pos[k][1]);
    scanf("%d", &test);
    while(test--) {
        cases++;
        for(i = 0; i < 4; i++) {
            for(j = 0; j < 4; j++) {
                scanf("%d", &k);
                init.board[i][j] = k;
                if(init.board[i][j] == 0) {
                    init.ix = i, init.iy = j;
                }
            }
        }
        if(solvable()) {
            solved = 0, initH = mxdep = H();
            if(!mxdep) {
                puts("");
                continue;
            }
            while(solved == 0)
                mxdep = IDA(0, initH, -1);
            //printf("%d\n", solved);
        }else {
            puts("This puzzle is not solvable.");
```

```
      }
   }
   return 0;
}
```

Output-



3.Queue

Program-

```
#include<stdio.h>
#include<string.h>

int main() {
   int T, N, P, R;
         long long DP[17][17][17];
         memset(DP, 0, sizeof(DP));

   DP[1][1][1] = 1;
   for(N = 2; N <= 13; N++)
      for(P = 1; P <= N; P++)
         for(R = 1; R <= N; R++)
            DP[N][P][R] = DP[N-1][P][R]*(N-2) + DP[N-1][P-1][R] + DP[N-1][P][R-1];
   scanf("%d", &T);
   while(T--) {
      scanf("%d %d %d", &N, &P, &R);
      printf("%lld\n", DP[N][P][R]);
   }
   return 0;

}
```

Output-

```
3
10 4 4
90720
11 3 1
1026576
3 1 2
1


...Program finished with exit code 0
Press ENTER to exit console.
```

4.Little Bishops

Program-

```
import java.io.*;
import java.util.*;
class Main
{
	static String ReadLn (int maxLg)  // utility function to read from stdin
	{
		byte lin[] = new byte [maxLg];
		int lg = 0, car = -1;

		try
		{
			while (lg < maxLg)
			{
				car = System.in.read();
				if ((car < 0) || (car == '\n')) break;
				lin [lg++] += car;
			}
		}
		catch (IOException e)
		{
			return (null);
		}

		if ((car < 0) && (lg == 0)) return (null);  // eof
		return (new String (lin, 0, lg));
```

```
}

public static void main (String args[])  // entry point from OS
{
	Main myWork = new Main();  // create a dinamic instance
	myWork.Begin();		// the true entry point
}

void Begin()
{
	StringTokenizer idata;
	String input;

	while ((input = Main.ReadLn (255)) != null){
		idata = new StringTokenizer (input);
		if(!idata.hasMoreTokens()){
			return;
		}
		int size = Integer.parseInt(idata.nextToken());
		int k = Integer.parseInt(idata.nextToken());
		if(k == 0 && size == 0){
			return;
		}
		littleBishops(size, k);
	}
	return;
}
static int count;
void littleBishops(int size, int k) {
	int sum = 0;
	if(size == 1 && k == 1){
		System.out.println(1);
		return;
	}
	if(k > size + size-2){
		System.out.println(0);
		return;
	}
	boolean[] even;
	boolean[] odd;
	if(size%2 == 0){
		even = new boolean[size-1];
		odd = new boolean[size];
	}
	else{
		even = new boolean[size];
		odd = new boolean[size-1];
	}
	for(int i = 0; i<k+1; i++){
		count = 0;
		numWays(even, odd, i, 1);
```

```
            int a = count;
            count = 0;
            numWays(even, odd, k-i, 0);
            int b = count;
            sum += a*b;
        }
        numWays(even, odd, k, 0);


        /*
        boolean[][] board = new boolean[size][size];
        numWays(board, k, 0, 0);
         */

        System.out.println(sum);
    }

    void numWays(boolean[] even, boolean[] odd, int k, int x) {
        if(k == 0){
            count++;
            return;
        }
        int middleo = (odd.length-1)/2;
        int middlee = (even.length)/2;
        int size = even.length + odd.length;
        for(int i = x; i<size-k+1; i+=2){
            int h = i;
            if(h > size/2){
                h = (size - h-1);
            }
            h = h/2;

            if(i%2 == 0){
                for(int j = middlee-h; j<middlee+h+1; j++){
                    if(!even[j]){
                        even[j] = true;
                        numWays(even, odd, k-1, i+2);
                        even[j] = false;
                    }
                }
            }
            if(i%2 == 1){
                for(int j = middleo-h; j<middleo+h+2; j++){
                    if(!odd[j]){
                        odd[j] = true;
                        numWays(even, odd, k-1, i+2);
                        odd[j] = false;
                    }
                }
            }
```

```
                /*
                for(int j = middle-h; j<middle+h+1; j+=2){
                        if(!column[j]){
                                boolean[] temp = column.clone();
                                temp[j] = true;
                                numWays(temp, k-1, i+1);
                        }
                }
                */
        }
    }
}
```

Output-