PLACEMENT PREPRATION

Problem 1: ACM ICPC Team

You are given a list of N people who are attending ACM-ICPC World Finals. Each of them are either well versed in a topic or they are not. Find out the maximum number of topics a 2-person team can know. And also find out how many teams can know that maximum number of topics.

Input Format
The first line contains two integers N and M separated by a single space, where N represents the number of people, and M represents the number of topics. N lines follow.
Each line contains a binary string of length M. In this string, 1 indicates that the i person knows a particular topic, and 0 indicates that the i person does not know the topic. Here, 1 ≤ i ≤ 2, and it denotes one of the persons in the team.

Output Format
On the first line, print the maximum number of topics a 2-people team can know.
On the second line, print the number of 2-person teams that can know the maximum number of topics.

Constraints
2 ≤ N ≤ 500
1 ≤ M ≤ 500

Sample Input
4 5
10101
11100
11010
00101
Sample Output
5
2

Explanation
(1, 3) and (3, 4) know all the 5 topics. So the maximal topics a 2-people team knows is 5, and only 2 teams can achieve this

```
import java.util.Scanner;

public class Solution {
        public static void main(String[] args) {
                        Scanner sc = new Scanner(System.in);
                        int n = sc.nextInt();  //4
```

```
            int m = sc.nextInt(); //5

            String[] topics = new String[n];
            topics[0] = sc.nextLine();
            System.out.println("top "+topics[0]);
            for(int i=0; i<n; i++){
                topics[i] = sc.nextLine();  //10101  11100 11010 00101
                if(topics[i].length() != m){
                    System.out.println("Run again");
                    System.exit(0);
                }
            }

            int maxTeam = 0;
            int maxSolution = 0;
            int checkSolution = 0;
            for(int i=0; i<n; i++)
            {
                for(int j=i; j<n; j++)
                {
                    if(i != j)
                    {
            checkSolution = countAns(topics[i],topics[j],m);  //10101  11100  5
                        if(checkSolution > maxSolution)
                        {
                            maxSolution = checkSolution;
                            maxTeam = 1;
                        }
                        else if(checkSolution == maxSolution)
                        {
                            maxTeam++;
                        }
                    }
                }
            }
            System.out.println(maxSolution);
            System.out.println(maxTeam);


     }

    private static int countAns(String str1, String str2, int m)  //10101  11100  5
        {
            int count = 0;
            int temp1,temp2;
            for(int i=0; i<m; i++){
                temp1 = str1.charAt(i) - '0';
                temp2 = str2.charAt(i) - '0';

                if((temp1 != 0) || (temp2 != 0)){
                    count++;
                }
                System.out.println(temp1 +" : "+temp2+" count :"+count);
            }

            return count;
        }

}
```

Program 2: Chocolate Feast

Little Bob loves chocolates, and goes to a store with in his pocket. The price of each chocolate is. The store offers a Discount: for every wrappers he gives to the store, he gets one chocolate for free. How many chocolates does Bob get to eat?

Input Format:
The first line contains the number of test cases .
T lines follow, each of which contains three integers N, C and M

Output Format:
Print the total number of chocolates Bob eats.

Constraints:
2 ≤ N ≤ 105
1 ≤ C ≤ N
2 ≤ M ≤ N

```
Sample input
3
10 2 5
12 4 4
6 2 2
Sample Output
6
3
5
```

Explanation

In the first case, he can buy 5 chocolates with and exchange the 5 wrappers to get one more chocolate. Thus, the total number of chocolates is 6.

In the second case, he can buy 3 chocolates for. However, it takes 4 wrappers to get one more chocolate. He can't avail the offer and hence the total number of chocolates remains 3.

In the third case, he can buy 3 chocolates for. Now he can give 2 of this 3 wrappers and get 1 chocolate. Again, he can use his 1 unused wrapper and 1 wrapper of new chocolate to get one more chocolate. So the total is 5

```
import java.util.Scanner;

public class SolutionChocolate
{
 public static void main(String[] args)
 {
           Scanner in = new Scanner(System.in);
           int t = in.nextInt();
           for(int i = 0; i < t; i++){
             System.out.println(Solve(in.nextInt(), in.nextInt(), in.nextInt()));
           }
         }

         private static long Solve(int N, int C, int M){   //10 2 5
           // N = rupees  10
           // c = one chocolate  price   2
           // m = num of wrapper for get offer  5
           int numCho = (int) Math.ceil(N/C);  //5
           int wrapper = numCho;//5
           while(wrapper >= M){
             wrapper -= M;   //
             wrapper++;
             numCho++;
           }

           return numCho;
         }
 }
```

Problem: Cut the sticks

You are given **N** sticks, where each stick is of positive integral length. A cut operation is performed on the sticks such that all of them are reduced by the length of the smallest stick.

Suppose we have 6 sticks of length

5 4 4 2 2 8

then in one cut operation we make a cut of length 2 from each of the 6 sticks. For next cut operation 4 sticks are left (of non zero length), whose length are

3 2 2 6

Above step is repeated till no sticks are left.

Given length of **N** sticks, print the number of sticks that are cut in subsequent cut operations.

Input Format

The first line contains a single integer N .

The next line contains N integers: a , a ,...a separated by space, where a represents the length of i stick.

Output Format

For each operation, print the number of sticks that are cut in separate line.

Constraints

1 ≤ N ≤ 1000

1 ≤ a ≤ 1000

Sample Input #00

6

5 4 4 2 2 8

Sample Output #00

6

4

2

1

Sample Input #01

8

1 2 3 4 3 3 2 1

Sample Output #01

8

6

4

1

Explanation

Sample Case #00 :

sticks-length l	ength-of-cut	sticks-cut
5 4 4 2 2 8	2	6
3 2 2 _ _ 6	2	4
1 _ _ _ _ 4	1	2

_ _ _ _ _ 3	3	1
_ _ _ _ _ _	DONE	DONE

Sample Case #01

sticks-length	length-of-cut	sticks-cut
1 2 3 4 3 3 2 1	1	8
_ 1 2 3 2 2 1 _	1	6
_ _ 1 2 1 1 _ _	1	4
_ _ _ 1 _ _ _ _	1	1
_ _ _ _ _ _ _ _	DONE	DONE

```
import java.util.Scanner;

public class CutStrickSolution {

   public static void main(String[] args) {
       /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class should be named
Solution. */
       Scanner sc = new Scanner(System.in);
       int number = sc.nextInt(); //6
       int[] stick = new int[number];
       for(int i=0; i<number; i++){
          stick[i] = sc.nextInt();  // 5 4 4 2 2 8
       }
       cut(stick);
   }
   public static void cut(int[] stick){
       int min=min(stick); // 5 4 4 2 2 8  //2
       int count=0;
       while(min != 0){
          count = 0;
          for(int i=0; i<stick.length; i++){
             if(stick[i] != 0){
                stick[i] -= min;   //3 2 2 0 0 6
                count++;
             }
          }
          System.out.println(count);
                            min = min(stick); //3 2 2 0 0 6
       };

   }
```

```
    public static int min(int[] stick){
       int min = getMaxValue(stick); //// 5 4 4 2 2 8  min = 8
       for(int i=0; i<stick.length; i++){
          if(stick[i] != 0 && stick[i] < min){
             min = stick[i];  //5 4 2
          }
       }
       return min; //2
    }

    public static int getMaxValue(int[] array){
                    int maxValue = array[0];  //// 5 4 4 2 2 8
                    for(int i=1;i < array.length;i++){
                              if(array[i] > maxValue){
                                        maxValue = array[i];
                              }
                    }
                    return maxValue; //8
           }
}
```

Problem: Box in a Box

Idea is to take a number as input and print a pattern of boxes If input is 2, two boxes are to be printed one inside the other Smallest box will be of size 3*3, the next bigger box will be 5*5, the next one will be 7*7, so on and so forth
For input 1, then draw a box of dimensions 3*3
For input 2, outer box will be 5*5, inner will be 3*3
For input 3, outer box will be 7*7, with 2 more inner boxes
So for n, outermost box will be **n*2 +1 i**n size, with (n1)
inner boxes
All boxes will be top left aligned as shown in the figure
Input Format:
First line of input contains a number N
Output Format:
Print N nested boxes

Constraints:

1. 0 < N < 25

Sample Input and Output

Example Number	Sample Input	Sample Output
1	2	* * * * * * * * * * * * * * * * * * *
2	3	* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Note:
Please do not use package and namespace in your code. For object oriented languages your code should be written in one
class.
Note:
Participants submitting solutions in C language should not use functions from <conio.h> / <process.h> as these files do not
exist in gcc
Note:
For C and C++, return type of main() function should be int.

```
mport java.util.InputMismatchException;
import java.util.Scanner;
```

```
public class BoxPattern
{
    public static void main(String[] args)
    {
        int N,row,count=0,j,i;
        Scanner sc=new Scanner(System.in);
        try
        {
            N=sc.nextInt();
            if(N>0 && N<25)
            {
                row=2*N+1;
                for(i=0;i<row;i++)
                {
                    if(i==0 || i==row-1)
                    {
                        for(j=0;j<row;j++)
                        {
                            System.out.print("*");
                        }
                    }
                    else if(i%2==1)
                    {
                        System.out.print("*");
                        for(j=1;j<=i;j++)
                        {
                            System.out.print(" ");
                        }
                        System.out.print("*");
                        for(j=i+2;j<row;j=j+2)
                        {
                            System.out.print(" *");
                        }
                    }
                    else
                    {
                        System.out.print("*");
                        for(j=1;j<=i;j++)
                        {
                            System.out.print("*");
                        }
                        for(j=i+2;j<row;j=j+2)
                        {
                            System.out.print(" *");
                        }
                    }
                    System.out.println();
                }
            }
            else
            {
                return;
            }
        }
        catch(InputMismatchException ex)
        {
            return;
        }

    }
}
```

Program: **Double and Add One Related to Prime Number.**

If you like numbers, you may have been fascinated by prime numbers.
Here's a problem related to prime numbers: Accept input numbers N and i. Identify all prime numbers P up to N with the
Following property:
P1=2*P+1 is also prime
P2=2*P1+1 is also prime
...
Pi=2*P (i1) +1 is also prime
Example: Inputs N=100, i=3

Let's start with p=2(it is also prime). Since i=3 we need 3 consecutive prime numbers that satisfy the Double and Add 1
Property explained below:
p1=2*p+1 translates to p1=2*2+1=5, which is prime
p2=2*p1+1 translates to p2=2*5+1=11, which is prime
p3=2*p2+1 translates to p3=2*11+1=23, which is prime
Hence p=2 is to be included in the output.
Next, if p=3, the derived numbers are 7, 15, 31 of which 15 is not prime. Hence p=3 is not a

Solution
Exploring other primes up to 100 in this fashion, we identify the following additional numbers to be included in the solution
For i=3:
5 (since the derived numbers 11, 23, 47 are all prime)
89 (since the derived numbers 179, 359, 719 are all prime)
Hence the output would be: 2, 5, 89

Input format for the example: 100 3
Output format for the example: 2 5 89
(Numbers separated by single space)
Input Format:
First line contains an integer N
Second line contains integer i
Output Format:
Space delimited prime numbers satisfying Double and Add 1 property in the given range N

Constraints:
1. 2< N <= 100000
2. 1< i <= 10
Sample Input and Output

Example Number	Sample Input	Sample Output
1	100 3	2 5 89
2	20 2	2 5 11

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class PrimeNumber
{
	public static void main(String[] args)
	{
		int N,j,i,flag=0,p,flag1=0;
		Scanner sc=new Scanner(System.in);
		try
		{
			N=sc.nextInt();

			if(N>2 && N<=100000)
			{
				i=sc.nextInt();
				if(i>1 && i<=10)
				{
					j=2;
					int x=0;
					p=j;
					while(j<=N && x<=i)
					{
						flag1=0;
```

```
if(p==2)
        flag1=1;
for(int t=2;t<=Math.sqrt(p);t++)
{
        if(p%t==0)
        {
                flag1=0;
                break;
        }

        else
                flag1=1;
}
flag=flag1;
if(flag==1)
{
                p=2*p+1;
                for(int t=2;t<=Math.sqrt(p);t++)
                {
                        if(p%t==0)
                        {
                                flag1=0;
                                break;
                        }

                        else
                                flag1=1;
                }
                flag=flag1;
                if(flag==1)
                {
                        x++;

                        if(x==i)
                        {
                                System.out.print(j+" ");
                                flag=0;
                                j++;
                                p=j;
                                x=0;
                        }
                }
}
else
{
```

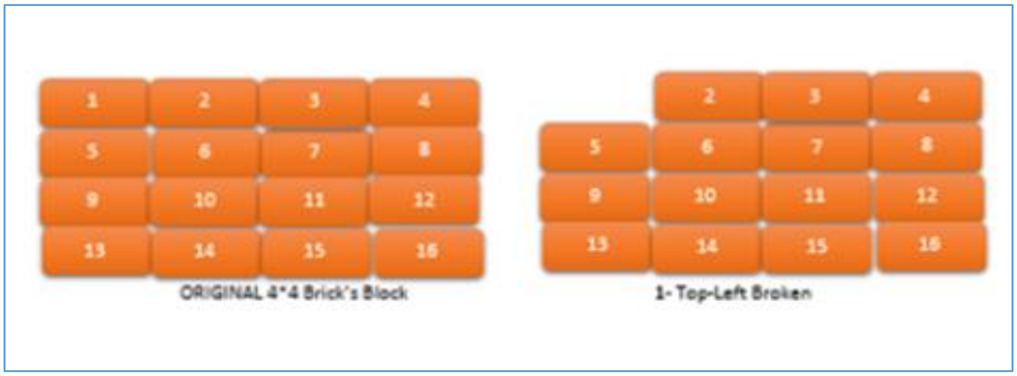
```
                            j++;
                            p=j;
                            x=0;
                        }
                    }
                }
                else
                {
                    return;
                }


            }
            else
            {
                return;
            }
        }
        catch(InputMismatchException  ex)
        {
            return;
        }

    }
}
```

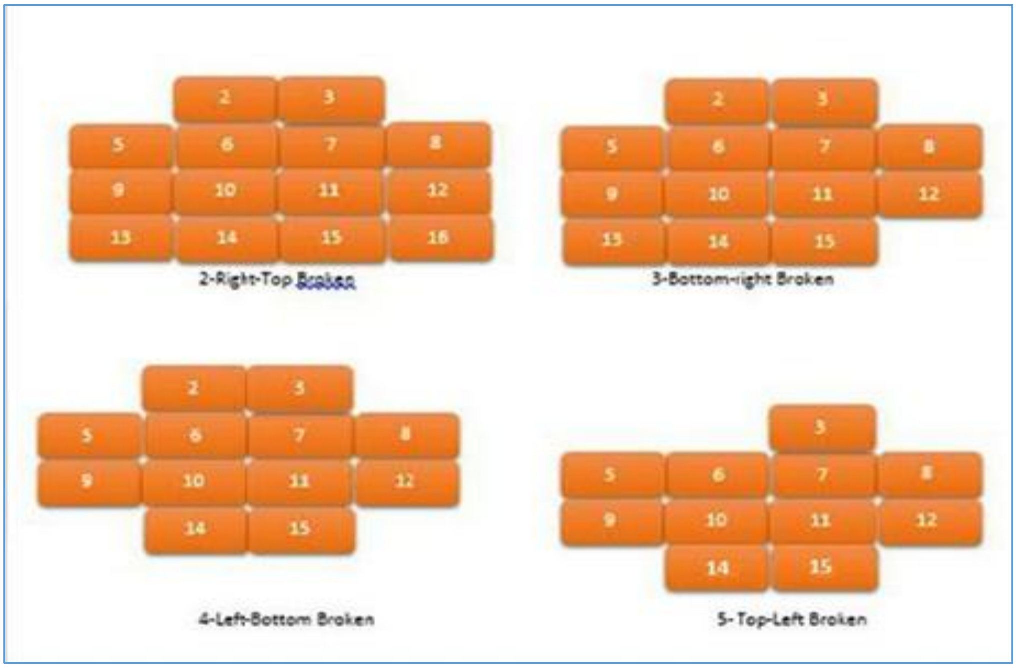
Problem: Clockwise Bricks Breaker

Clockwise Bricks Breaker is a game where there is a ball which is used to break the block of bricks. Brick's Blocks are always in a form of a square matrix(e.g 2X2, 4X4). The ball has some characteristics like for breaking each brick from the block. It consumes 1 unit of energy and the whole setup is in a manner that the ball will break bricks always in clock wise direction, in a ring fashion and start from top left.
Take an example of 4X4 brick block.



ORIGINAL 4*4 Brick's Block

1- Top-Left Broken

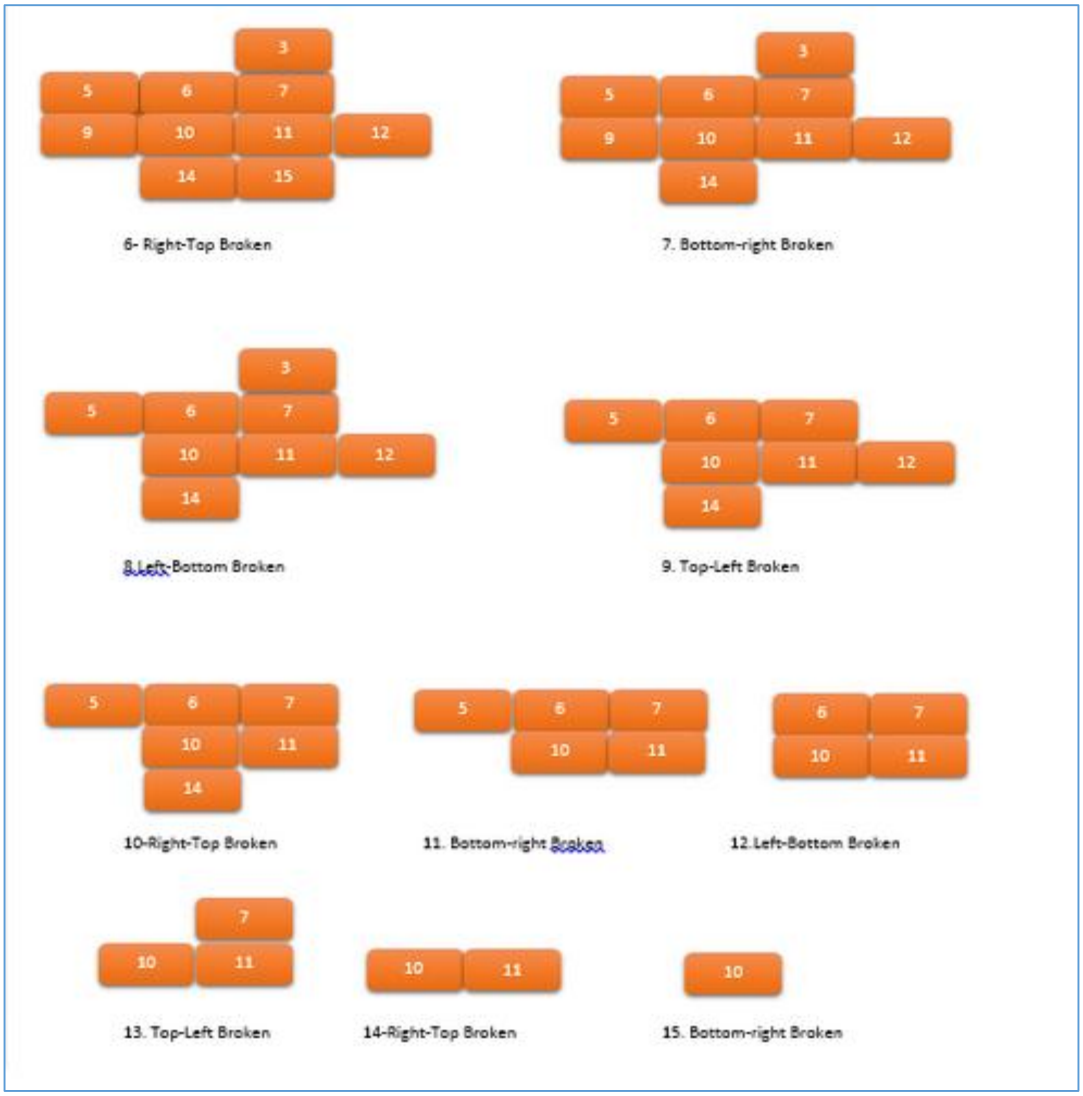


2-Right-Top Broken

3-Bottom-right Broken

4-Left-Bottom Broken

5-Top-Left Broken



6- Right-Top Broken

7. Bottom-right Broken

8.Left-Bottom Broken

9. Top-Left Broken

10-Right-Top Broken

11. Bottom-right Broken

12.Left-Bottom Broken

13. Top-Left Broken

14-Right-Top Broken

15. Bottom-right Broken

The numbers on the bricks are nothing but points gained by breaking that brick which is assigned row wise starting from 1 to NxN. The game gets over when energy of ball exhausts and once ball initiated with some energy it will only stop when all its energy exhaust.

Your Task is to calculate total points (points of a block should be the positional value of that block as mentioned in the picture: You have to add points of all the broken bricks) gained by a player, where the energy of ball and size of block will be given.

Input Format:
First line of input should be the number of test cases T
Next T lines contain two variables N and E delimited by a whitespace where,

N =size of square block (i.e. block will be of NXN bricks)
E =Energy of ball

Output Format:
Output will be T lines containing the integer which is total number points gained by that energy.
OR
Print "Invalid Input" if constraint fails
Constraints:
0< T<=10
0< N <=200
0<= E <=N*N

Example Number	Sample Input	Sample Output	Explaination
1	24 5 1 50	36 Invalid Input	For first test case as with Energy 5, the ball will break 5 bricks having points 1,4,16,13,2 So 1+4+16+13+2=36. Second test case is Invalid Input because constraint E<=N*N not satisfied.
2	25 0 9 3	0 Invalid Input	For first test case it is 0 as energy is zero. For second test case it is Invalid Input because Constraint E >= 0 not satisfied.

```
package com.sdj;

import java.util.InputMismatchException;
import java.util.Scanner;

public class BricksBraker
{
        public static void main(String[] args)
        {
                int N,T,E;
                Scanner sc=new Scanner(System.in);
                try
                {
                T=sc.nextInt();
                if(T>0 && T<=10)
                {
                        for(int y=0;y<T;y++)
                                {
                                N=sc.nextInt();
                                if(N>0 && N<=200)
```

```
{
E=sc.nextInt();
if(E>=0 && E<=N*N)
{
    int k=N-1,x=0,sum=0;
    int num[]=new int[N*N];
    for(int j=0;k>=0; j++)
    {
        if(k==0)
        {
        num[x]=1+j*(N+1);
        k--;
        }
        else
        {
            for(int i=1;i<=k; i++)
            {
            num[x++]=i+j*(N+1);
            num[x++]=(N*i)+j*(N-1);
            num[x++]=(N*N)-i+1-j*(N+1);
        num[x++]=N*(N-i)+1-j*(N-1);
            if(i==k)
                {
                    k=k-2;
                    break;
                }

                }
            }

            }
            for(int i=0;i<E;i++)
            {
                sum=sum+num[i];

            }
            System.out.println(sum);
        }
        else
        {
            System.out.println("Invalid Input");
        }
    }
    else
    {
        System.out.println("Invalid Input");
```

```
                        }
                    }

                }
                else
                {
                    System.out.println("Invalid Input");
                    return;
                }
            }
            catch(InputMismatchException ex)
            {
                return;
            }

        }

    }
```

Problem : Utopian Tree

The Utopian tree goes through 2 cycles of growth every year. The first growth cycle occurs during the monsoon, when it doubles in height. The second growth cycle occurs during the summer, when its height increases by 1 meter.

Now, a new Utopian tree sapling is planted at the onset of the monsoon. Its height is 1 meter. Can you find the height of the tree after N growth cycles?

Input Format

The first line contains an integer, T , the number of test cases.

T lines follow. Each line contains an integer, N , that denotes the number of cycles for that test case

Constraints

1 <= T <= 10

0 <= N <= 60

Output Format

For each test case, print the height of the Utopian tree after N cycles.

Sample Input #00:

```
2
0
1
```

Sample Output #00:

```
1
```

2

Explanation #00:
There are 2 test cases. When N = 0, the height of the tree remains unchanged. When N = 1, the tree doubles its height as it's
Planted just before the onset of monsoon.

Sample Input: #01:
2
3
4
Sample Output: #01:
6
7
Explanation: #01:
There are 2 testcases.
N = 3:
the height of the tree at the end of the 1 cycle = 2
the height of the tree at the end of the 2 cycle = 3
the height of the tree at the end of the 3 cycle = 6
N = 4:
the height of the tree at the end of the 4 cycle = 7

```
public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int testCase = sc.nextInt();
        int num = 0;
        int height ;
        for(int i=0; i<testCase; i++){
            num = sc.nextInt();
            height = 1;
            for(int j=0; j<num; j++){
                if(j%2 == 0){
                    height *= 2;
                }else{
                    height += 1;
                }
            }
            System.out.println(height);
        }
    }
}
```

Problem: The Love-Letter Mystery

James found a love letter his friend Harry has written for his girlfriend. James is a prankster, so he decides to meddle with the letter. He changes all the words in the letter into palindromes.
To do this, he follows 2 rules:
(a) He can reduce the value of a letter, e.g. he can change 'd' to 'c', but he cannot change 'c' to 'd'.
(b) In order to form a palindrome, if he has to repeatedly reduce the value of a letter, he can do it until the letter becomes 'a'.
Once a letter has been changed to 'a', it can no longer be changed.
Each reduction in the value of any letter is counted as a single operation. Find the minimum number of operations required to
convert a given string into a palindrome.

Input Format
The first line contains an integer T , i.e., the number of test cases.
The next T lines will contain a string each.
Output Format
A single line containing the number of minimum operations corresponding to each test case.

Constraints
1 ≤ T ≤ 10
1 ≤ length of string ≤ 10
All characters are lower case English letters.
Sample Input #00

```
3
abc
abcba
abcd
```

Sample Output #00

```
2
0
4
```

Explanation
For the first test case, ab*c* -> ab*b* -> ab*a*.
For the second test case, abcba is a palindromic string.
For the third test case, abc*d* -> abc*c* -> abc*b* -> abc*a* = ab*c*a -> ab*b*a.

```
package com.sdj;
import java.util.Scanner;
```

```
public class LoveLetter Mystery
{
        public static void main(String[] args)
        {

        Scanner sc = new Scanner(System.in);
        int testCase =sc.nextInt();
        while(0 < testCase){
                int count = 0;
                String str = sc.next();  //abcd
                str = str.toLowerCase(); //abcd
                int len = str.length(); //4
                int halfLen = (int) Math.round(len/2.0); //2
                for(int i=0; i<halfLen; i++){
                        if(str.charAt(i) != str.charAt(len-i-1)){
                        count =count +  (int) Math.abs(str.charAt(i) - str.charAt(len-i-1));  //a - d
                        System.out.println( (int) Math.abs(str.charAt(i) - str.charAt(len-i-1)));

                        }
                }
                System.out.println(count);
                testCase--;
        }
        }
}
```

Problem : Sherlock and the Beast

Sherlock Holmes is getting paranoid about Professor Moriarty, his archenemy. All his efforts to subdue Moriarty have been in vain. These days Sherlock is working on a problem with Dr. Watson. Watson mentioned that the CIA has been facing weird problems with their supercomputer, 'The Beast', recently.

This afternoon, Sherlock received a note from Moriarty, saying that he has infected 'The Beast' with a virus. Moreover, the note had the number N printed on it. After doing some calculations, Sherlock figured out that the key to remove the virus is the largest 'Decent' Number having N digits.

A 'Decent' Number has –

1. Only 3 and 5 as its digits.
2. Number of times 3 appears is divisible by 5.
3. Number of times 5 appears is divisible by 3.

Meanwhile, the counter to destruction of 'The Beast' is running very fast. Can you save 'The Beast', and find the key before Sherlock?

Input Format

The 1st line will contain an integer T , the number of test cases. This is followed by T lines, each containing an integer N i.e. the Number of digits in the number

Output Format

Largest Decent number having N digits. If no such number exists, tell Sherlock that he is wrong and print '-1'

Constraints

1<=T <=20
1<=N <=100000

Sample Input

```
4
1
3
5
11
```

Sample Output

```
-1
555
33333
55555533333
```

Explanation

For N =1, there is no such number.
For N =3, 555 is only possible number.
For N =5, 33333 is only possible number.
For N =11, 55555533333 and all of permutations of digits are valid numbers, among them, the given number is the largest one.

```
package codevita;

import java.util.Scanner;

public class Beast
{
	public static void main(String[] args) {
	    Scanner sc =  new Scanner(System.in);
```

```
            int testCase = sc.nextInt();//1
            while(testCase > 0){
                int number = sc.nextInt();  //5
                String str="";
                for(int i=number; i>=0; i--){  //5//4//3
                    if(i%3 == 0 && (number-i)%5 == 0){
                        str = repeat("5", i) + repeat("3", number-i);
                        break;
                    }
                }
                str = str.length() == 0 ? "-1" : str;
                System.out.println(str);
                testCase--;
            }
        }

        public static String repeat(String str,int times){
            StringBuilder temp = new StringBuilder();
            for(int i=0; i<times; i++){
                temp.append(str);
            }
            return temp.toString();
        }
}
```

Program: Pangrams

Roy wanted to increase his typing speed for programming contests. So, his friend advised him to type the sentence "The quick brown fox jumps over the lazy dog" repeatedly because it is a pangram. (Pangrams are sentences constructed by using every letter of the alphabet at least once.)

After typing the sentence several times, Roy became bored with it. So he started to look for other pangrams.

Given a sentence, tell Roy if it is a pangram or not.

Input Format

Input consists of a line containing.

Constraints

Length of can be atmost 10^3 ($1 \le |s| \le 10^3$) and it may contain spaces, lowercase and uppercase letters. Lowercase and Uppercase instances of a letter are considered same.

Output Format
Output a line containing pangram if is a pangram, otherwise output not pangram.

Sample Input #1
We promptly judged antique ivory buckles for the next prize

Sample Output #1

pangram

Sample Input #1
We promptly judged antique ivory buckles for the prize

Sample Output #1

not pangram

Explanation
In the first test case, the answer is pangram because the sentence contains all the letters.

```
package com.javapadho;

import java.util.Scanner;

public class Pangram
{
        public static void main(String[] args) {
           Scanner sc = new Scanner(System.in);

                        String str = sc.nextLine();
                        str = str.toLowerCase();
                        int len = str.length();
                        String temp = " ";

                        if(len > 25){
                                for(int i=0; i<len; i++){
                                        if(temp.length() == 27){
```

```
                        break;
                    }

                    if(temp.indexOf(str.charAt(i)) == -1){
                        temp = temp + str.charAt(i);
                        System.out.println(temp);
                    }

                }
            }

            String answer = (temp.length() == 27) ? "pangram" : "not pangram";

            System.out.println(answer);
        }
}
```

Program 1: Bubble Sort.

```
class Sort
{
    public static int[] bubbleSort(int a[])
    {
        for(int pass=a.length-1;pass>=0;pass--)
        {
            for(int i=0;i<=pass-1;i++)
            {
                if(a[i]>a[i+1])
                {
                    int temp  = a[i];
                    a[i] = a[i+1];
                    a[i+1] = temp;
                }
            }
        }
        return a;
    }
    public static void display(int a[])
    {
        for(int i=0;i<a.length;i++)
        {
            System.out.println(a[i]);
        }
    }
}
public class ManagerBubbleSort
```

```
{
  public static void main(String[] args)
  {
                                          int a[]={10,30,20,25,65,78};
                                          Sort.bubbleSort(a);
                                          Sort.display(a);


 }
}
```

Program 2: Bubble sort using Swap variable.

```
public class ManagerBubbleSort1
{
          public static int[] bubbleSort(int a[])
           {
                    int swapped=1;
                     for(int pass=a.length-1;pass>=0&&swapped>=0;pass--)
                     {
                              swapped=0;
                               for(int i=0;i<=pass-1;i++)
                               {
                                       if(a[i]>a[i+1])
                                       {
                                                   int temp  = a[i];
                                                   a[i] = a[i+1];
                                                   a[i+1] = temp;
                                                   swapped=1;
                                       }
                               }
                     }
                     return a;
           }
           public static void display(int a[])
           {
                     for(int i=0;i<a.length;i++)
                     {
                              System.out.println(a[i]);
                     }
           }
           public static void main(String[] args)
            {
                                                   int a[]={10,30,20,25,65,78};
```

```
                                            bubbleSort(a);
                                            display(a);


          }
}

import java.util.Scanner;

/*public class GCD1
{
    public static int gcd(int m,int n)   //m= 2525    n=25
    {
           int gcd=1;
            if(m==n)
            {
                  return m;
            }
            for(int k=n/2;k>=1;k--)  //   k=12  11
            {
                   if(m%k==0&&n%k==0) //m=2525%12 &&25%12
                   {
                            gcd = k;
                            break;
                   }
            }
            return gcd;
    }
    public static void main(String[] args)
    {
                    Scanner sc = new Scanner(System.in);
                    System.out.println("enter first integer");
                    int m  = sc.nextInt();
                    System.out.println("enter second integer");
                    int n = sc.nextInt();
                    System.out.println("The greatest common divisor for "+m+"and "+n+"is "+gcd(m,n));

          }
}
*/
public class GCD1
{
    public static int gcd(int m,int n)   //m= 2525    n=25
    {
            if(m%n==0)
```

```
        {
            return n;
        }
        else
        {
            return gcd(n,m%n);
        }
    }
    public static void main(String[] args)
    {
              Scanner sc = new Scanner(System.in);
              System.out.println("enter first integer");
              int m  = sc.nextInt();
              System.out.println("enter second integer");
              int n = sc.nextInt();
              System.out.println("The greatest common divisor for "+m+"and "+n+"is "+gcd(m,n));

        }
}
```

PrimeNumber.java

```
import java.util.Scanner;

public class PrimeNumber
{
  public static void main(String[] args)
  {
        Scanner sc = new Scanner(System.in);
        System.out.println("Find all prime number <=n enter n");
        int n  = sc.nextInt();
        final int NUMBER_PER_LINE=10;
        int count=0;
        int number=2;
        System.out.println("The prime numbers are :");
        while(number<=n)
        {
            boolean isPrime =true;
            for(int divisor=2;divisor<=(int)(Math.sqrt(number));divisor++)
                        {
                            if(number%divisor==0)
                            {
                                isPrime=false;
                                break;
                            }
```

```
                }
        if(isPrime)
        {
            count++;

            if(count%NUMBER_PER_LINE==0)
            {
                System.out.printf("%7d\n",number);
            }
            else
            {
                System.out.printf("%7d",number);
            }

        }
        number++;

    }

 }
}
```

EfficientPrimeNumber.java

```
package dsacareermonk;

import java.util.Scanner;

public class EfficientPrimeNumbers
{
 public static void main(String[] args)
 {
            Scanner sc = new Scanner(System.in);
        System.out.println("Find all prime number <=n enter n");
         int n  = sc.nextInt();
         //A list to hold prime number
         java.util.List<Integer> list = new java.util.ArrayList<Integer>();
         final int NUMBER_PER_LINE=10; //Display 10 per line
         int count=0;  //Count the number of prime number
         int number=2; // A number to test primeness
         int squareRoot=1; //
         System.out.println("The prime numbers are :");
         while(number<=n)
         {
            boolean isPrime =true;
             if(squareRoot*squareRoot<number)
                 squareRoot++;
```

```
                         //Test if number is prime

                         for(int k=0;k<list.size()&&list.get(k)<=squareRoot;k++)
                         {
                                 if(number%list.get(k)==0)
                                 {
                                         isPrime =false;
                                         break;
                                 }

                         }
                        if(isPrime)
                        {
                                 count++;
                                 list.add(number);
                                 if(count%NUMBER_PER_LINE==0)
                                 {
                                       System.out.println(number);
                                 }
                                 else
                                 {
                                       System.out.print(number+" ");
                                 }
                        }
                        number++;

                 }

 }
}
```

Prime number using **SieveOfEratosthenes** Algorithm.

```
public class SieveEratosthenes
{
  public static void main(String[] args)
  {
                Scanner sc = new Scanner(System.in);
                System.out.println("Find all prime  number <=n ,enter n:");
                int n  = sc.nextInt();
                boolean primes[] = new boolean[n+1]; //Prime number sieve
                for(int i=0;i<primes.length;i++)
                {
                        primes[i] = true;
                }
                for(int k=2;k<=n/k;k++)
                {
```

```
            if(primes[k])
            {
                for(int i=k;i<=n/k;i++)
                {
                    primes[k*i] = false;  // k*i is not  prime
                }
            }
        }
        final int  NUMBER_PER_LINE=10;
        int count=0;
        for(int i=2;i<primes.length;i++)
        {
            if(primes[i])
            {
                count++;
                if(count%10==0)
                {
                    System.out.printf("%7d\n",i);
                }
                else
                {
                   System.out.printf("%7d\n",i);
                }
            }
        }
        System.out.println("\n "+count+"number of primes "+n);
 }
}
```