Take N as input. Print the sum of its odd placed digits and sum of its even placed digits.

Input Format

Constraints

```
0 < N <= 1000000000
```

Output Format

Sample Input

```
2635
```

Sample Output

```
11
5
```

Explanation

5 is present at 1st position, 3 is present at 2nd position, 6 is present at 3rd position and 2 is present at 4th position.

Sum of odd placed digits on first line. 5 and 6 are placed at odd position. Hence odd place sum is 5+6=11

Sum of even placed digits on second line. 3 and 2 are placed at even position. Hence even place sum is 3+2=5

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int count=0;
        int even=0;
        int odd=0;
        while(n>=1){
            if(count%2==0){
                odd+=n%10;
            }
            else{
                even+=n%10;
            }
            n/=10;
            count++;
        }
        System.out.println(odd);
        System.out.println(even);
    }
}
```

Print "lowercase" if user enters a character between 'a-z' , Print "UPPERCASE" is character lies between 'A-Z' and print "Invalid" for all other characters like $,.^# etc.

Input Format

Single Character .

Constraints

```
-
```

Output Format

lowercase UPPERCASE Invalid

Sample Input

```
$
```

Sample Output

```
Invalid
```

Explanation

```
-
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        char c = sc.next().charAt(0);
        int value=0;

        for(int i =65; i<=90; i++){
            if((int)(c)==i){
                System.out.println("UPPERCASE");
                value=1;
                break;
            }
            else if((int)(c)==i+32){
                System.out.println("lowercase");
                value=1;
                break;
            }
        }
        if(value != 1)
            System.out.println("Invalid");
    }
}
```

Take N as input. For a value of N=7, we wish to draw the following pattern :

```
      1
    2 1   1 2
  3 2 1       1 2 3
4 3 2 1           1 2 3 4
  3 2 1       1 2 3
    2 1   1 2
      1
```

Input Format

Take N as input.

Constraints

N is odd number.

Output Format

Pattern should be printed with a space between every two values.

Sample Input

```
7
```

Sample Output

```
      1
    2 1   1 2
  3 2 1       1 2 3
4 3 2 1           1 2 3 4
  3 2 1       1 2 3
    2 1   1 2
      1
```

Explanation

Catch the pattern and print it accordingly.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        int row = 1;
        int star = 1;
        int space = n-1;
        int count=1;
        int mspace = -1;

        while(row<=n){
            int i = 1;
            while(i<=space){
                System.out.print("  ");
                i++;
            }
            int j = 1;
            while(j<=star){
                System.out.print(count+" ");
                j++;
                count--;
            }
            int k = 1;
            while(k<=mspace){
                System.out.print("  ");
                k++;
            }
            count++;
            int l = 1;
            while(l<=star){
                if((row!=n && row!=1)||l!=star)
                    System.out.print(count+" ");
                l++;
                count++;
            }
            if(row<n/2+1){
            star++;
            space-=2;
            mspace+=2;
            }
            else{
                count-=2;
                star--;
                space+=2;
                mspace-=2;
            }
            System.out.println();
            row++;
        }
    }
}
```

Take N (number in binary format). Write a function that converts it to decimal format and Print the value returned.

Input Format

Constraints

```
0 < N <= 1000000000
```

Output Format

Sample Input

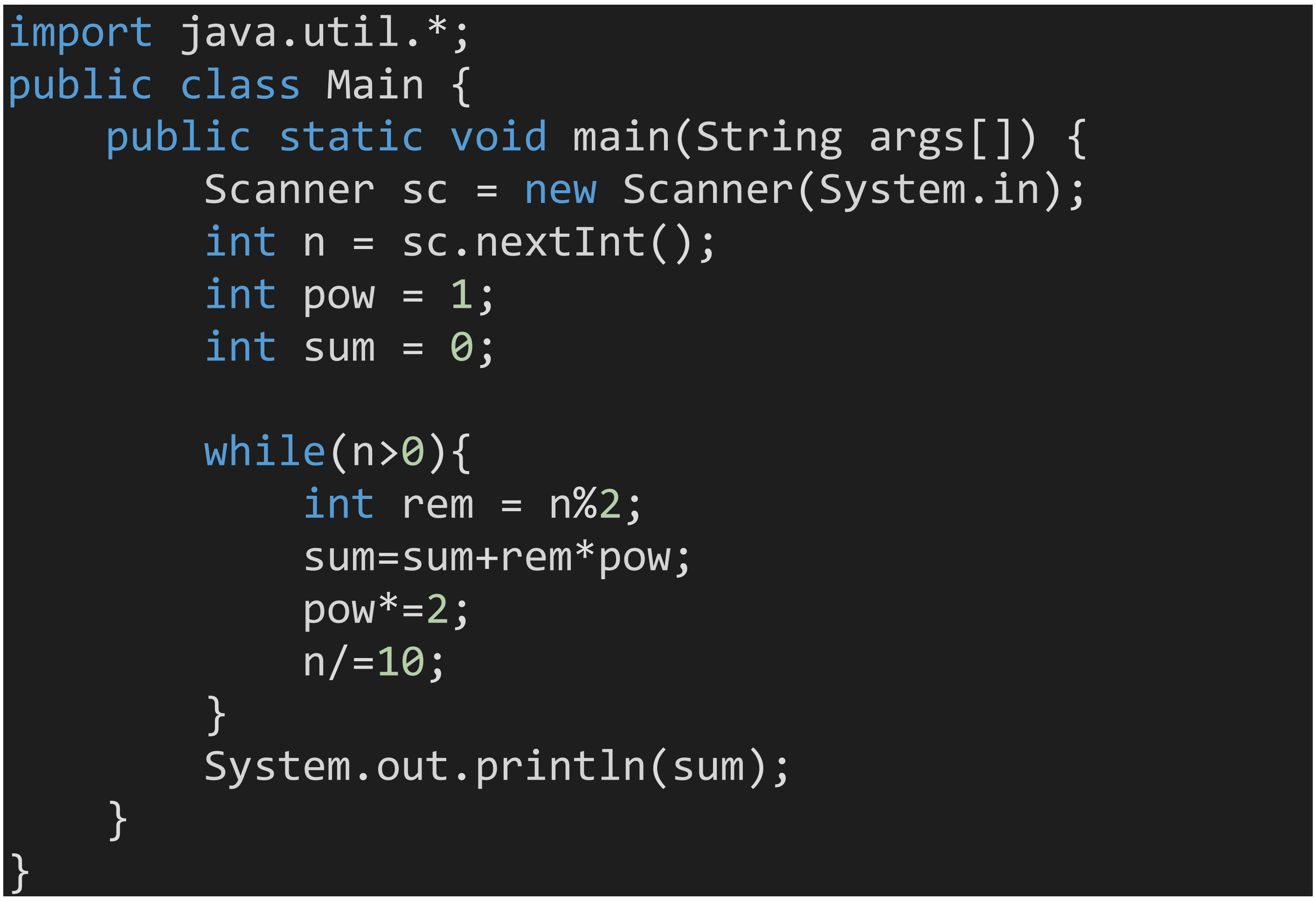
```
101010
```

Sample Output

```
42
```

Explanation

For binary number fedcba , Decimal number = f * 2^5 + e * 2^4 + d * 2^3 + ...+ a * 2^0.



```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int pow = 1;
        int sum = 0;

        while(n>0){
            int rem = n%2;
            sum=sum+rem*pow;
            pow*=2;
            n/=10;
        }
        System.out.println(sum);
    }
}
```

Take N (number of rows - only odd numbers allowed), print the following pattern (for N = 5).

```
		*
	*	*	*
*	*	*	*	*
	*	*	*
		*
```

Input Format

Constraints

```
0 < N < 10 (only odd numbers allowed)
```

Output Format

Sample Input

```
5
```

Sample Output

```
    *
  * * *
* * * * *
  * * *
    *
```

Explanation

```
Each '*' is separated from other by a tab.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int row = 1 ;
        int space = n/2;
        int star = 1;
        while(row<=n) {
            int i=1;
            while(i<=space) {
                System.out.print("  ");
                i++;
            }
            int j=1;
            while(j<=star) {
                System.out.print("* ");
                j++;
            }
            int k=1;
            while(k<=space) {
                System.out.print("  ");
                k++;
            }
            if(row<n/2+1) {
                space--;
                star+=2;
            }
            else {
                space++;
                star-=2;
            }
            row++;
            System.out.println();
        }
    }
}
```

Given a binary number ,help Von Neuman to find out its decimal representation. For eg 000111 in binary is 7 in decimal.

Input Format

The first line contains N , the number of binary numbers. Next N lines contain N integers each representing binary represenation of number.

Constraints

```
N<=1000 Digits in binary representation is <=16.
```

Output Format

N lines,each containing a decimal equivalent of the binary number.

Sample Input

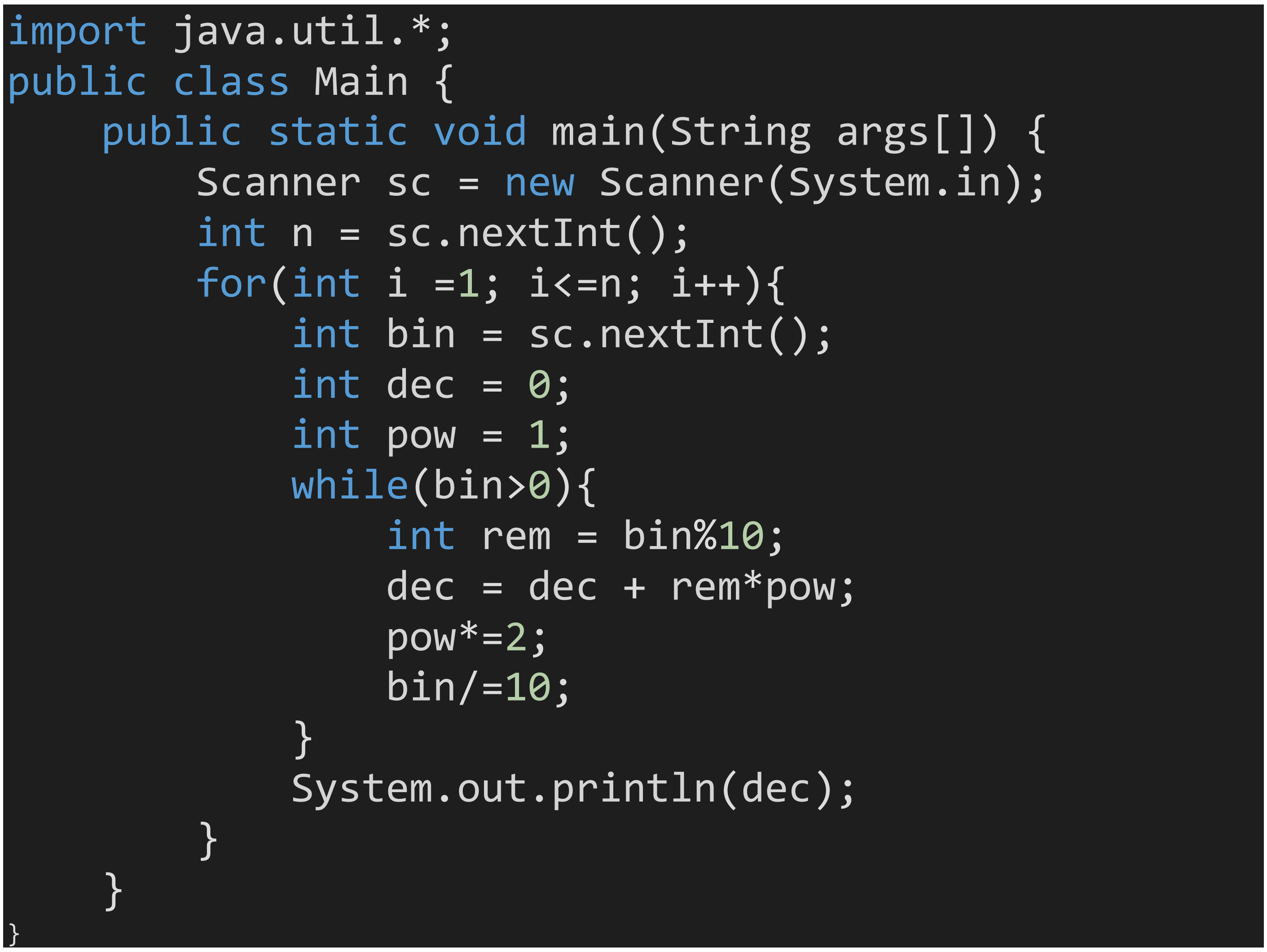
```
4
101
1111
00110
111111
```

Sample Output

```
5
15
6
63
```

Explanation

For binary number fedcba , Decimal number = f * 2^5 + e * 2^4 + d * 2^3 + ...+ a * 2^0.



```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for(int i =1; i<=n; i++){
            int bin = sc.nextInt();
            int dec = 0;
            int pow = 1;
            while(bin>0){
                int rem = bin%10;
                dec = dec + rem*pow;
                pow*=2;
                bin/=10;
            }
            System.out.println(dec);
        }
    }
}
```

Take N (number of rows), print the following pattern (for N = 4)

1
2 3
4 5 6
7 8 9 10

Input Format

Constraints

0 < N < 100

Output Format

Sample Input

```
4
```

Sample Output

```
1
2	3
4	5	6
7	8	9	10
```

Explanation

Each number is separated from other by a tab.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int row = 1;
        int star = 1;
        int count = 1;

        while(row<=n){
            int i = 1;
            while(i<=star){
                System.out.print(count+"    ");
                i++;
                count++;
            }
            row++;
            star++;
            System.out.println();
        }
    }
}
```

Take N (number of rows), print the following pattern (for N = 5)
1
2 2
3 0 3
4 0 0 4
5 0 0 0 5

Input Format

There will be an integer in input.

Constraints

```
0 < N < 100
```

Output Format

Print the pattern.

Sample Input

```
5
```

Sample Output

```
1
2	2
3	0	3
4	0	0	4
5	0	0	0	5
```

Explanation

Each number is separated from other by a tab.If row number is n (>1), total character is n. First and last character is n and rest are 0.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc =  new Scanner(System.in);
        int n = sc.nextInt();
        int row = 1;
        int star = 1;

        while(row <= n){
            int i = 1;
            while(i<=star){
                if(i==1 || i==star)
                    System.out.print(row+"   ");
                else
                    System.out.print(0+"  ");
                i++;
            }
            star++;
            row++;
            System.out.println();
        }
    }
}
```

Take the following as input.

Minimum Fahrenheit value
Maximum Fahrenheit value
Step

Print as output the Celsius conversions. Use the formula C = (5/9)(F – 32) E.g. for an input of 0, 100 and 20 the output is
0 -17
20 -6
40 4
60 15
80 26
100 37

Input Format

The first line of the input contains an integer denoting the Minimum Fahrenheit value. The second line of the input contains an integer denoting the Maximum Fahrenheit value. The third line of the input contains an integer denoting the Step.

Constraints

```
0 < Min < 100 Min < Max < 500 0 < Step
```

Output Format

Print Fahrenheit and Celsius values separated by a tab. Each step should be printed in a new line.

Sample Input

```
0
100
20
```

Sample Output

```
0 -17
20 -6
40 4
60 15
80 26
100 37
```

Explanation

```
First number in every output line is fahrenheit, second number
is celsius. The two numbers are separated by a tab.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
    Scanner sc = new Scanner(System.in);
    int min = sc.nextInt();
    int max = sc.nextInt();
    int dif = sc.nextInt();

    for(int i=min; i<=max; i+=dif){
        int c = (int)(5*(i-32)/9);
        System.out.println(i+"  "+c);
    }
    }
}
```

Take as input a number N, print "Prime" if it is prime if not Print "Not Prime".

Input Format

Constraints

2 < N <= 1000000000

Output Format

Sample Input

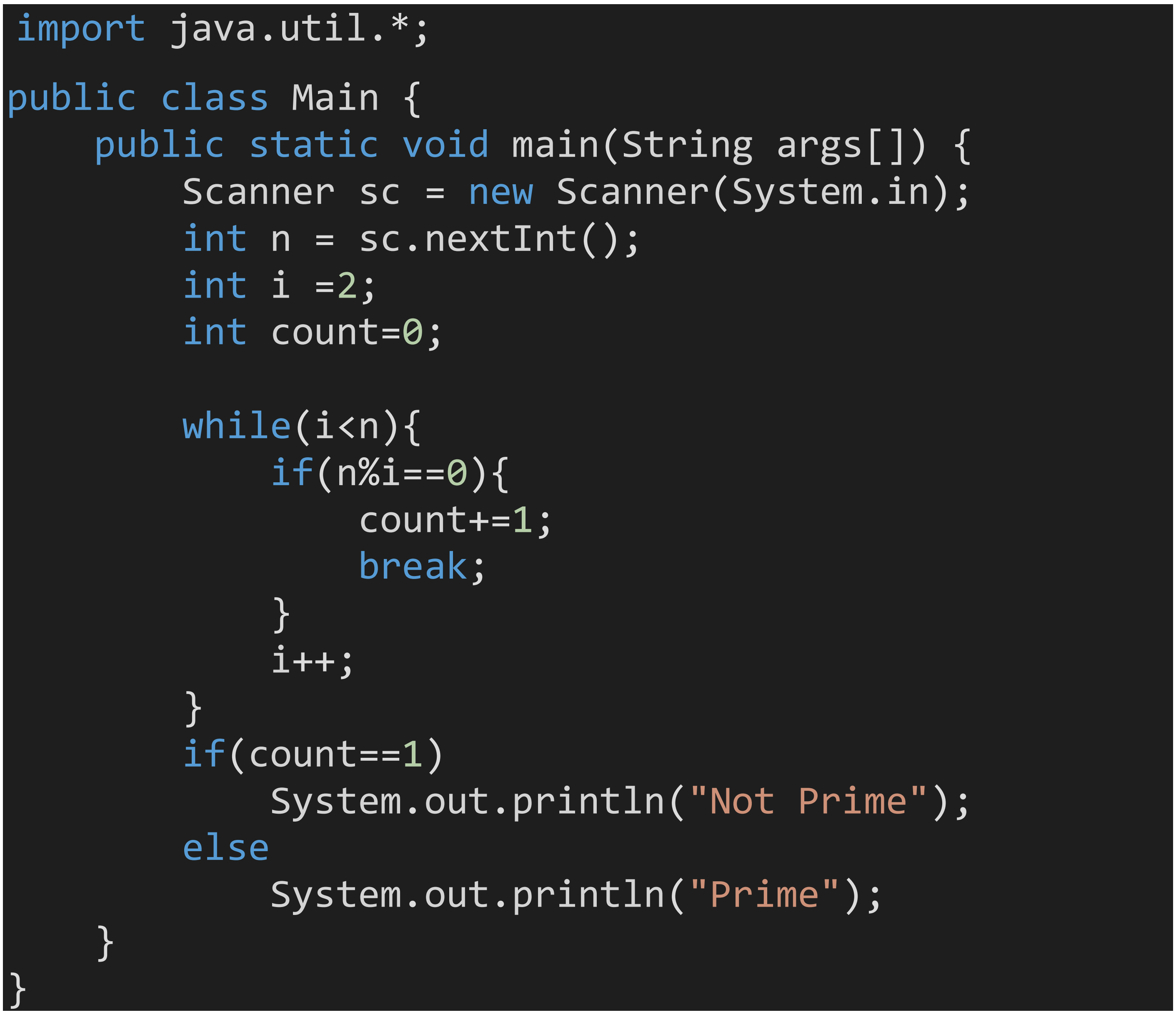
```
3
```

Sample Output

```
Prime
```

Explanation

The output is case specific



```java
import java.util.*;

public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int i =2;
        int count=0;

        while(i<n){
            if(n%i==0){
                count+=1;
                break;
            }
            i++;
        }
        if(count==1)
            System.out.println("Not Prime");
        else
            System.out.println("Prime");
    }
}
```

Luke Skywalker gave Chewbacca an integer number x. Chewbacca isn't good at numbers but he loves inverting digits in them. Inverting digit t means replacing it with digit 9 - t.

Help Chewbacca to transform the initial number x to the minimum possible positive number by inverting some (possibly, zero) digits. The decimal representation of the final number shouldn't start with a zero.

Input Format

The first line contains a single integer x ($1 \le x \le 10^{18}$) — the number that Luke Skywalker gave to Chewbacca.

Constraints

```
x <= 1000000000000000000
```

Output Format

Print the minimum possible positive number that Chewbacca can obtain after inverting some digits. The number shouldn't contain leading zeroes.

Sample Input

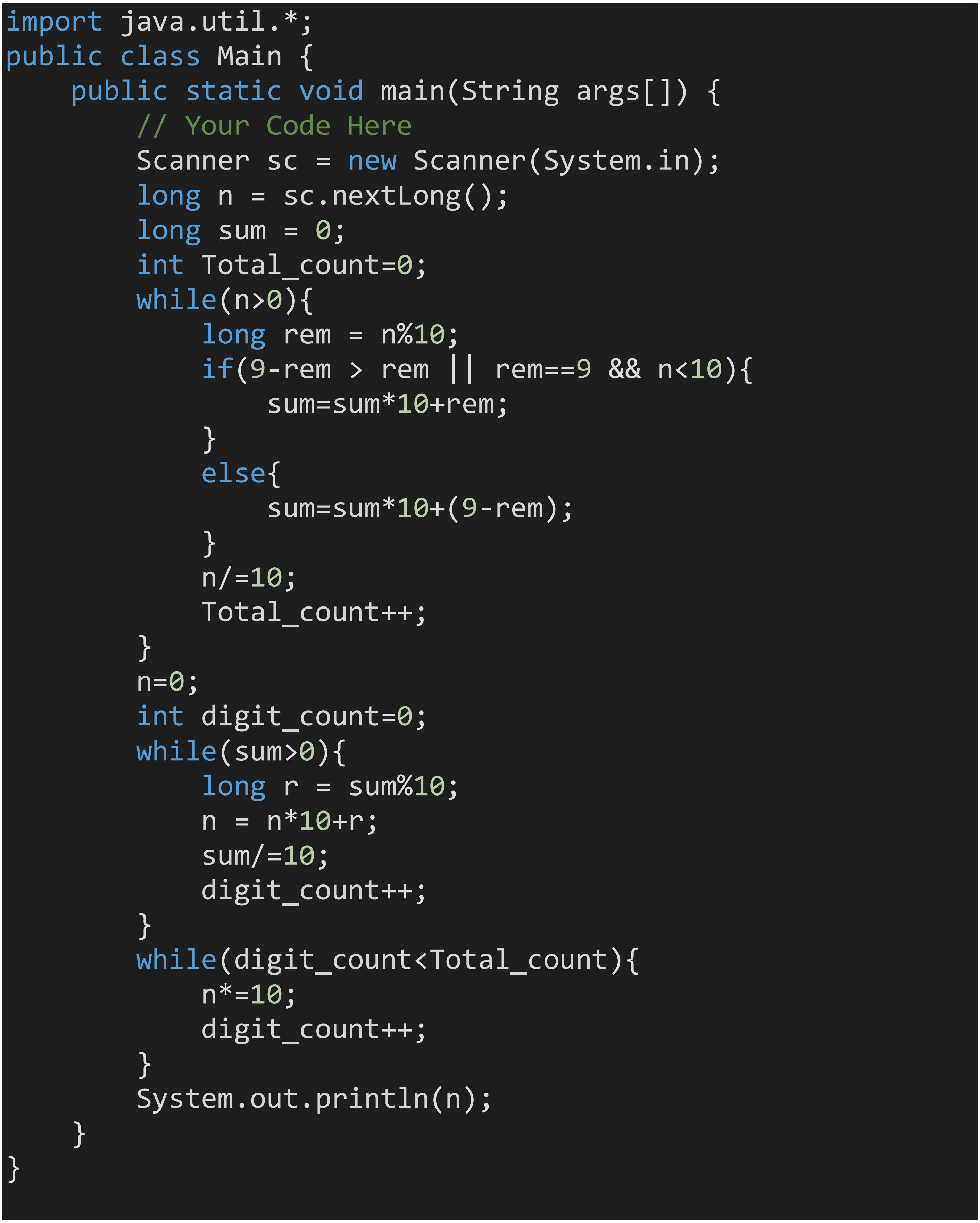
```
4545
```

Sample Output

```
4444
```

Explanation

```
There are many numbers form after inverting the digit. For
minimum number, check if inverting digit is less than or
greater than the original digit. If it is less, then invert it
otherwise leave it.
```



```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        long n = sc.nextLong();
        long sum = 0;
        int Total_count=0;
        while(n>0){
            long rem = n%10;
            if(9-rem > rem || rem==9 && n<10){
                sum=sum*10+rem;
            }
            else{
                sum=sum*10+(9-rem);
            }
            n/=10;
            Total_count++;
        }
        n=0;
        int digit_count=0;
        while(sum>0){
            long r = sum%10;
            n = n*10+r;
            sum/=10;
            digit_count++;
        }
        while(digit_count<Total_count){
            n*=10;
            digit_count++;
        }
        System.out.println(n);
    }
}
```

Take the following as input.

A number (N1)
A number (N2)
Write a function which prints first N1 terms of the series 3n + 2 which are not multiples of N2.

Input Format

Constraints

```
0 < N1 < 100 0 < N2 < 100
```

Output Format

Sample Input

```
10
4
```

Sample Output

```
5
11
14
17
23
26
29
35
38
41
```

Explanation

```
The output will've N1 terms which are not divisible by N2.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n1 = sc.nextInt();
        int n2 = sc.nextInt();
        int count = 1;
        int n = 1;
        while(count<=n1){
            int num = 3*n+2;
            if(num%n2!=0){
                System.out.println(num);
                count++;
            }
            n++;
        }
    }
}
```

Take N (number in decimal format). Write a function that converts it to octal format. Print the value returned.

Input Format

Constraints

```
0 < N <= 1000000000
```

Output Format

Sample Input

```
63
```

Sample Output

```
77
```

Explanation

```
Both input and output are integers
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int pow = 1;
        int sum = 0;

        while(n>0){
            int rem = n%8;
            sum=sum+rem*pow;
            pow*=10;
            n/=8;
        }
        System.out.println(sum);
    }
}
```

Take sb (source number system base), db (destination number system base) and sn (number in source format). Write a function that converts sn to its counterpart in destination number system. Print the value returned.

Input Format

Constraints

```
0 < N <= 1000000000
```

Output Format

Sample Input

```
8
2
33
```

Sample Output

```
11011
```

Explanation

```
All input output is as integers and in separate lines.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int sb = sc.nextInt();
        int db = sc.nextInt();
        int num = sc.nextInt();
        if(sb==10 || db == 10){
            int pow=1;
            int dnum = 0;
            while(num>0){
                int r = num%db;
                dnum=dnum+r*pow;
                pow*=sb;
                num/=db;
            }
            System.out.println(dnum);
        }
        else{
            int mb=10;
            int pow=1;
            int mnum = 0;
            while(num>0){
                int r = num%mb;
                mnum=mnum+r*pow;
                pow*=sb;
                num/=mb;
            }
            pow=1;
            int dnum=0;
            while(mnum>0){
                int r = mnum%db;
                dnum=dnum+r*pow;
                pow*=mb;
                mnum/=db;
            }
            System.out.println(dnum);
         }
    }  }
```

A Boston number is a composite number, the sum of whose digits is the sum of the digits of its prime factors obtained as a result of prime factorization (excluding 1). The first few such numbers are 4,22 ,27 ,58 ,85 ,94 and 121 . For example, 378 = 2 × 3 × 3 × 3 × 7 is a Boston number since 3 + 7 + 8 = 2 + 3 + 3 + 3 + 7. Write a program to check whether a given integer is a Boston number.

Input Format

There will be only one line of input:N , the number which needs to be checked.

Constraints

```
1 < N < 2,147,483,647 (max value of an integer of the size of 4
bytes)
```

Output Format

1 if the number is a Boston number. 0 if the number is a not Boston number.

Sample Input

```
378
```

Sample Output

```
1
```

Explanation

```
Self Explanatory
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int n1=n;
        int i = 2;
        int psum = 0;
        while(i<=n){
            while(n%i==0){
                psum = psum + sumofdigit(i);
                n/=i;
            }
            i++;
        }
        int sum = sumofdigit(n1);
        if(psum==sum)
            System.out.println(1);
        else
            System.out.print(0);
        n=n1;
    }
        public static int sumofdigit(int n){
            int sum = 0;
            while(n>0){
                int rem = n%10;
                sum=sum+rem;
                n/=10;
            }
            return sum;
        }
}
```

Take N (number of rows), print the following pattern (for N = 4)
0
1 1
2 3 5
8 13 21 34

Input Format

Constraints

```
0 < N < 100
```

Output Format

Sample Input

```
4
```

Sample Output

```
0
1       1
2       3       5
8      13      21      34
```

Explanation

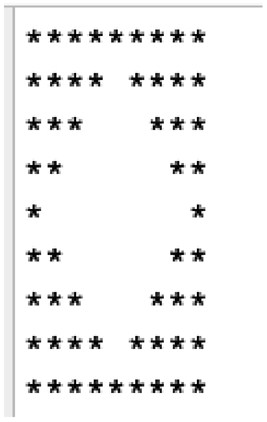
Each number is separated from other by a tab. For given input n, You need to print n(n+1)/2 fibonacci numbers. Kth row contains , next k fibonacci numbers.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int row = 1;
        int star = 1;
        int a = 0;
        int b = 1;

        while(row<=n){
            int i =1;
            while(i<=star){
                System.out.print(a+"  ");
                int c = a+b;
                a=b;
                b=c;
                i++;
            }
            row++;
            star++;
            System.out.println();
        }
    }
}
```

You will be given a number N. You have to code a hollow diamond looking pattern.

The output for N=5 is given in the following image.



```
*********
**** ****
***   ***
**     **
*       *
**     **
***   ***
**** ****
*********
```

Input Format

The input has only one single integer N.

Constraints

Output Format

Output the given pattern.

Sample Input

Sample Output

Explanation

Write a code to print above given pattern. No space between any two characters.

```java
import java.util.*;
public class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        int row = 1;
        int star = n;
        int space = -1;

        while(row<=2*n-1){
            int i = 1;
            while(i<=star){
                System.out.print("*");
                i++;
            }
            int j = 1;
            while(j<=space){
                System.out.print(" ");
                j++;
            }
            int k = 1;
            while(k<=star){
                if((row != 2*n-1 && row!=1) || k!=star)
                    System.out.print("*");
                k++;
            }
            if(row<n){
                star--;
                space+=2;
            }
            else{
                star++;
                space-=2;
            }
            row++;
            System.out.println();
        }
    }
}
```

Due to an immense rise in Pollution, Kejriwal is back with the Odd and Even Rule in Delhi. The scheme is as follows, each car will be allowed to run on Sunday if the sum of digits which are even is divisible by 4 or sum of digits which are odd in that number is divisible by 3. However to check every car for the above criteria can't be done by the Delhi Police. You need to help Delhi Police by finding out if a car numbered N will be allowed to run on Sunday?

Input Format

The first line contains N , then N integers follow each denoting the number of the car.

Constraints

```
N<=1000 Car No >=0 && Car No <=1000000000
```

Output Format

N lines each denoting "Yes" or "No" depending upon whether that car will be allowed on Sunday or Not !

Sample Input

```
2
12345
12134
```

Sample Output

```
Yes
No
```

Explanation

```
1 + 3 + 5 = 9 which is divisible by 3
1 + 1 + 3 = 5 which is NOT divisible by 3 and 2+4 = 6 which is
not divisble by 4.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        for(int i = 1; i<=n;i++){
            int input = sc.nextInt();
            int odd = 0;
            int even = 0;

            while(input>0){
                int rem = input%10;
                if(rem%2==0)
                    even+=rem;
                else
                    odd+=rem;
                input/=10;
            }
            if(odd%3==0)
                System.out.println("Yes");
            else if(even%4==0)
                System.out.println("Yes");
            else
                System.out.println("No");
        }
    }
}
```

Take the following as input.

A number
Write a function which returns true if the number is an armstrong number and false otherwise, where Armstrong number is defined as follows.

A positive integer of n digits is called an Armstrong number of order n (order is number of digits) if.

abcd... = pow(a,n) + pow(b,n) + pow(c,n) + pow(d,n) + ...

1634 is an Armstrong number as 1634 = 1^4 + 6^4 + 3^4 + 4^4

371 is an Armstrong number as 371 = 3^3 + 7^3 + 1^3

Input Format

Single line input containing an integer

Constraints

```
0 < N < 1000000000
```

Output Format

Print boolean output for each testcase.
"true" if the given number is an Armstrong Number, else print "false".

Sample Input

```
371
```

Sample Output

```
true
```

Explanation

```
Use functions. Write a function to get check if the number is
armstrong number or not. Numbers are armstrong if it is equal
to sum of each digit raised to the power of number of digits.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int n1 = n;
        int count=0;
        int sum = 0;

        while(n>0){
            count+=1;
            n/=10;
        }
        n=n1;

        while(n1>0){
            int rem = n1%10;
            int prod = 1;
            int i = 1;
            while(i<=count){
                prod*=rem;
                i++;
            }
            sum+=prod;
            n1/=10;
        }
        if(n==sum)
            System.out.print(true);
        else
            System.out.print(false);
    }
}
```

Given coefficients of a quadratic equation , you need to print the nature of the roots (Real and Distinct , Real and Equal or Imaginary) and the roots.
If Real and Distinct , print the roots in increasing order.
If Real and Equal , print the same repeating root twice
If Imaginary , no need to print the roots.

Note : Print only the integer part of the roots.

Input Format

First line contains three integer coefficients a,b,c for the equation ax^2 + bx + c = 0.

Constraints

```
-100 <= a, b, c <= 100
```

Output Format

Output contains one/two lines. First line contains nature of the roots .The next line contains roots(in non-decreasing order) separated by a space if they exist. If roots are imaginary do not print the roots. Output the integer values for the roots.

Sample Input

```
1 -11 28
```

Sample Output

```
Real and Distinct
4 7
```

Explanation

```
The input corresponds to equation ax^2 + bx + c = 0. Roots =
(-b + sqrt(b^2 - 4ac))/2a , (-b - sqrt(b^2 - 4ac))/2a
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        int c = sc.nextInt();
        int d=(b*b-4*a*c);
        int root_value=0;
        if(d<0){
            root_value=0;
            System.out.println("Imaginary");
        }
        else if(d==0){
            root_value=1;
            System.out.println("Real and Equal");
        }
        else{
            root_value=2;
            System.out.println("Real and Distinct");
        }
        if(root_value==1 || root_value==2){
            int root1 = (int)((-b + Math.sqrt(d))/2*a);
            int root2 = (int)((-b - Math.sqrt(d))/2*a);
            if(root1<=root2){
                System.out.print(root1+" ");
                System.out.print(root2);
            }
            else{
                System.out.print(root2+" ");
                System.out.print(root1);
            }
        }
    }
}
```

Take the following as input.
A number
A digit
Write a function that returns the number of times digit is found in the number. Print the value returned.

Input Format

Integer (A number) Integer (A digit)

Constraints

```
0 <= N <= 1000000000 0 <= Digit <= 9
```

Output Format

Integer (count of times digit occurs in the number)

Sample Input

```
5433231
3
```

Sample Output

```
3
```

Explanation

The digit can be from 0 to 9. Assume decimal numbers.In the given case digit 3 is occurring 3 times in the given number.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int num = sc.nextInt();
        int digit = sc.nextInt();
        int count = 0;
        while(num>0){
            int rem = num % 10;
            if(rem == digit)
                count++;
            num/=10;
        }
        System.out.println(count);
    }
}
```

Take N as input, Calculate it's reverse also Print the reverse.

Input Format

Constraints

```
0 <= N <= 1000000000
```

Output Format

Sample Input

```
123456789
```

Sample Output

```
987654321
```

Explanation

You've to calculate the reverse in a number, not just print the reverse.

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        // Your Code Here
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int rev = 0;
        while(n>0){
            int rem = n%10;
            rev=rev*10+rem;
            n/=10;
        }
        System.out.println(rev);
    }
}
```

Take N (number of rows), print the following pattern (for N = 4).

```
      1
    2 3 2
  3 4 5 4 3
4 5 6 7 6 5 4
```

Input Format

Constraints

```
0 < N < 10
```

Output Format

Sample Input

```
4
```

Sample Output

```
			1
		2	3	2
	3	4	5	4	3
4	5	6	7	6	5	4
```

Explanation

```
Each number is separated from other by a tab.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        int row=1;
        int star=1;
        int space=n-1;
        int count=1;

        while(row<=n) {
            int i =1;
            while(i<=space) {
                System.out.print("  ");
                i++;
            }
            int j =1;
            while(j<=star) {
                    System.out.print(count+" ");
                    if(j<row){
                        count++;
                    }
                    else{
                        count--;
                    }
                j++;
            }
            count+=2;
            row++;
            star+=2;
            space--;
            System.out.println();
        }

    }
}
```

Given a list of numbers, stop processing input after the cumulative sum of all the input becomes negative.

Input Format

A list of integers to be processed

Constraints

```
All numbers input are integers between -1000 and 1000.
```

Output Format

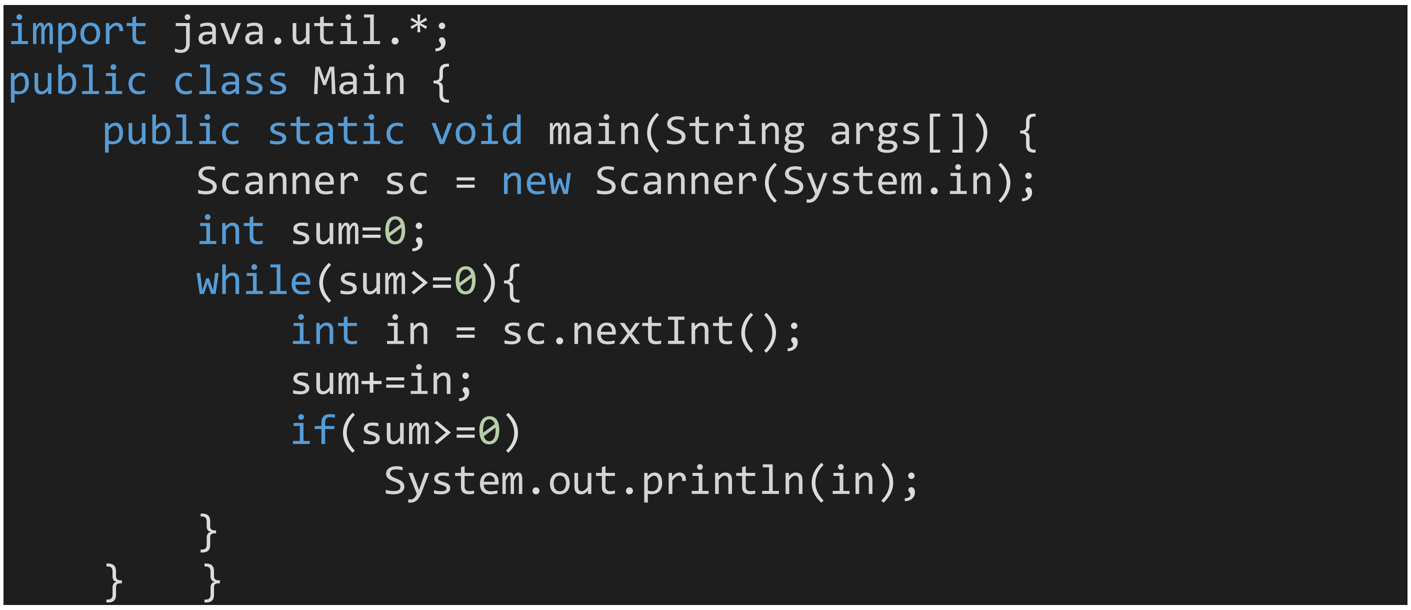
Print all the numbers before the cumulative sum become negative.

Sample Input

```
1
2
88
-100
49
```

Sample Output

```
1
2
88
```



```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int sum=0;
        while(sum>=0){
            int in = sc.nextInt();
            sum+=in;
            if(sum>=0)
                System.out.println(in);
        }
    }   }
```

Take the following as input.

A number (N1)
A number (N2)
Write a function which returns the LCM of N1 and N2. Print the value returned.

Input Format

Constraints

```
0 < N1 < 1000000000 0 < N2 < 1000000000
```

Output Format

Sample Input

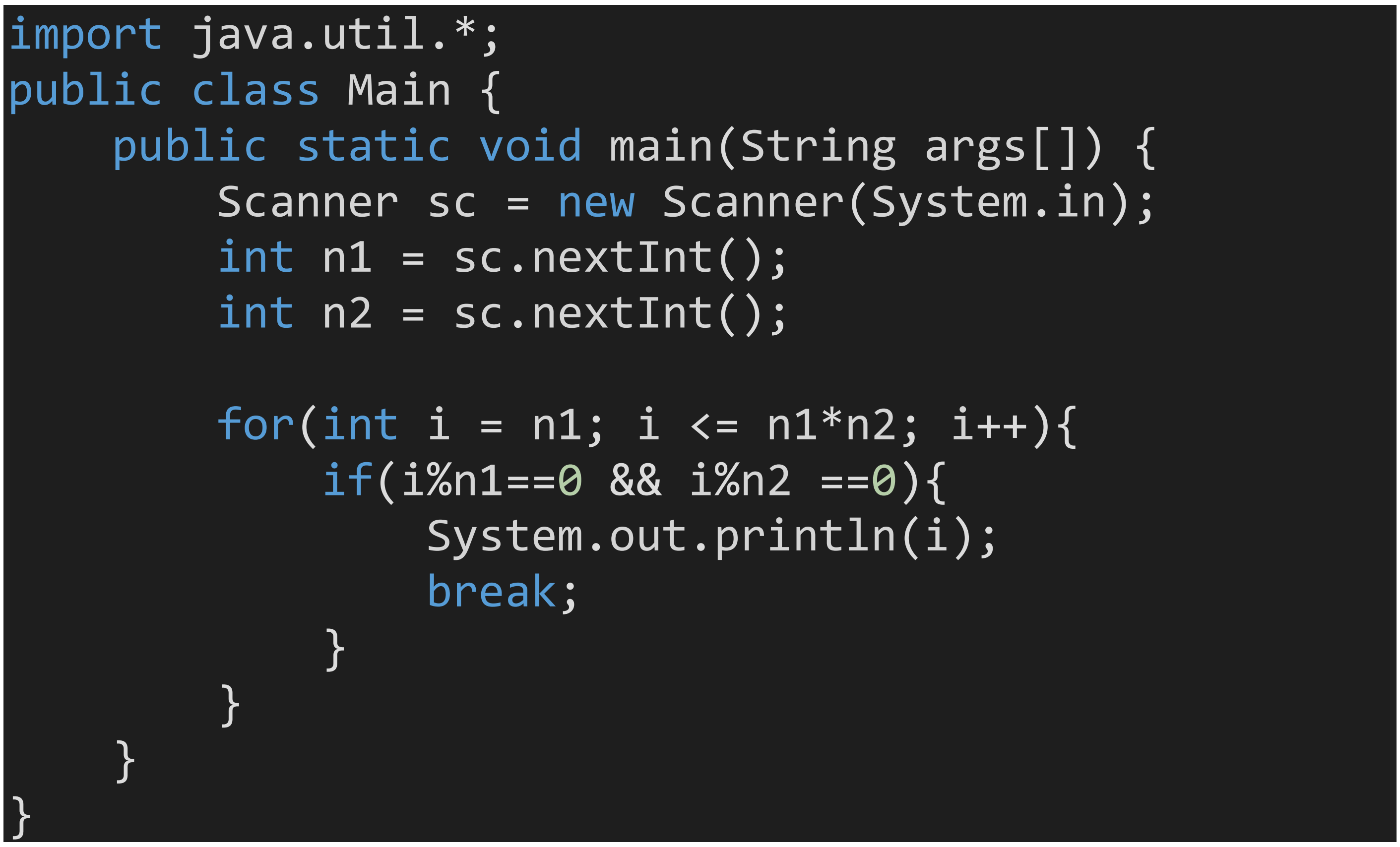
```
4
6
```

Sample Output

```
12
```

Explanation

```
The smallest number that is divisible by both N1 and N2 is
called the LCM of N1 and N2.
```



```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n1 = sc.nextInt();
        int n2 = sc.nextInt();

        for(int i = n1; i <= n1*n2; i++){
            if(i%n1==0 && i%n2 ==0){
                System.out.println(i);
                break;
            }
        }
    }
}
```

Given number of rows N, you have to print a Hollow Rhombus. See the output for corresponding given input.

Input Format

Single integer N.

Constraints

```
N <= 20
```

Output Format

Print pattern.

Sample Input

```
5
```

Sample Output

```
    *****
   *   *
  *   *
 *   *
*****
```

Explanation

```
For any input N. First line contains 4 space and then 5 {*} and
then the second line contains according to the output
format.
```

```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int star = n;
        int space = n-1;
        int row = 1;

        while(row<=n){
            int i =1;
            while(i<=space){
                System.out.print(" ");
                i++;
            }
            int j = 1;
            while(j<=star){
                if(j==1 || j==star || row==1 ||
row==n)
                    System.out.print("*");
                else
                    System.out.print(" ");
                j++;
            }
            row++;
            space--;
            System.out.println();
        }
    }
}
```

Take N as input. Print Nth Fibonacci Number, given that the first two numbers in the Fibonacci Series are `0` and `1`.

Input Format

Constraints

```
0 <= N <= 1000
```

Output Format

Sample Input

```
10
```

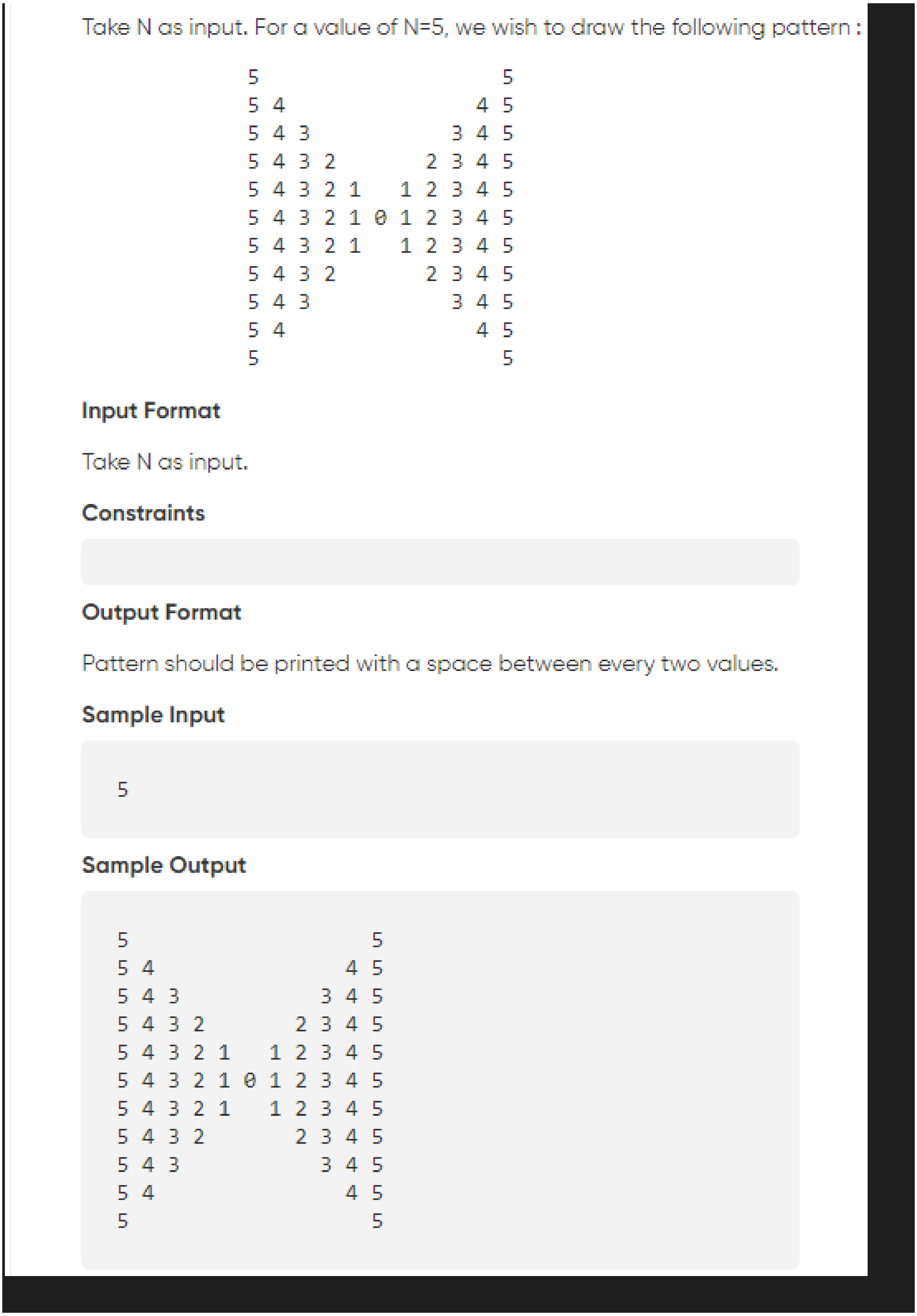
Sample Output

```
55
```

Explanation

```
The 0th fibonnaci is 0 and 1st fibonnaci is 1.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int a = 0;
        int b = 1;
        for(int i=1; i<=n; i++) {
            int c=a+b;
            a=b;
            b=c;
        }
        System.out.println(a);
    }
}
```


Take N as input. For a value of N=5, we wish to draw the following pattern :

```
5                   5
5 4               4 5
5 4 3           3 4 5
5 4 3 2       2 3 4 5
5 4 3 2 1   1 2 3 4 5
5 4 3 2 1 0 1 2 3 4 5
5 4 3 2 1   1 2 3 4 5
5 4 3 2       2 3 4 5
5 4 3           3 4 5
5 4               4 5
5                   5
```

Input Format

Take N as input.

Constraints

Output Format

Pattern should be printed with a space between every two values.

Sample Input

```
5
```

Sample Output

```
5                   5
5 4               4 5
5 4 3           3 4 5
5 4 3 2       2 3 4 5
5 4 3 2 1   1 2 3 4 5
5 4 3 2 1 0 1 2 3 4 5
5 4 3 2 1   1 2 3 4 5
5 4 3 2       2 3 4 5
5 4 3           3 4 5
5 4               4 5
5                   5
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc= new Scanner(System.in);
        int n = sc.nextInt();
        int row=1;
        int star=1;
        int space=2*n-1;
        int count = n;

        while(row<=2*n+1){
            int i = 1;
            while(i<=star){
                System.out.print(count+" ");
                count--;
                i++;
            }
            count++;
            int j = 1;
            while(j<=space){
                System.out.print("  ");
                j++;
            }
            int k = 1;
            while(k<=star){
                if(row!=n+1 || k!=1)
                    System.out.print(count+" ");
                count++;
                k++;
            }
            if(row<=n){
                star++;
            }
            else{
                star--;
                space+=2;
            }
            count--;
            row++;
            System.out.println();
        }
    }
}
```

Take the following as input.
A number
Assume that for a number of n digits, the value of each digit is from 1 to n and is unique. E.g. 32145 is a valid input number.

Write a function that returns its inverse, where inverse is defined as follows

Inverse of 32145 is 12543. In 32145, "5" is at 1st place, therefore in 12543, "1" is at 5th place; in 32145, "4" is at 2nd place, therefore in 12543, "2" is at 4th place.

Print the value returned.

Input Format

Integer

Constraints

```
0 < N < 1000000000
```

Output Format

Integer

Sample Input

```
32145
```

Sample Output

```
12543
```

Explanation

```
Assume that for a number of n digits, the value of each digit
is from 1 to n and is unique. E.g. 32145 is a valid input number.
Inverse of 32145 is 12543. In 32145, "5" is at 1st place, therefore
in 12543, "1" is at 5th place; in 32145, "4" is at 2nd place,
therefore in 12543, "2" is at 4th place.
```

```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int sum = 0;
        int count=1;
        while(n>0){
            int rem = n%10;
            sum=(int)(count*Math.pow(10,rem-1));
            n/=10;
            count+=1;
        }
        System.out.println(sum);
    }
}
```

Take the following as input.

A number (N1)
A number (N2)
Write a function which returns the GCD of N1 and N2. Print the value returned.

Input Format

Two integers seperated by a new line.

Constraints

```
0 < N1 < 1000000000
0 < N2 < 1000000000
```

Output Format

Output a single integer which is the GCD of the given integers.

Sample Input

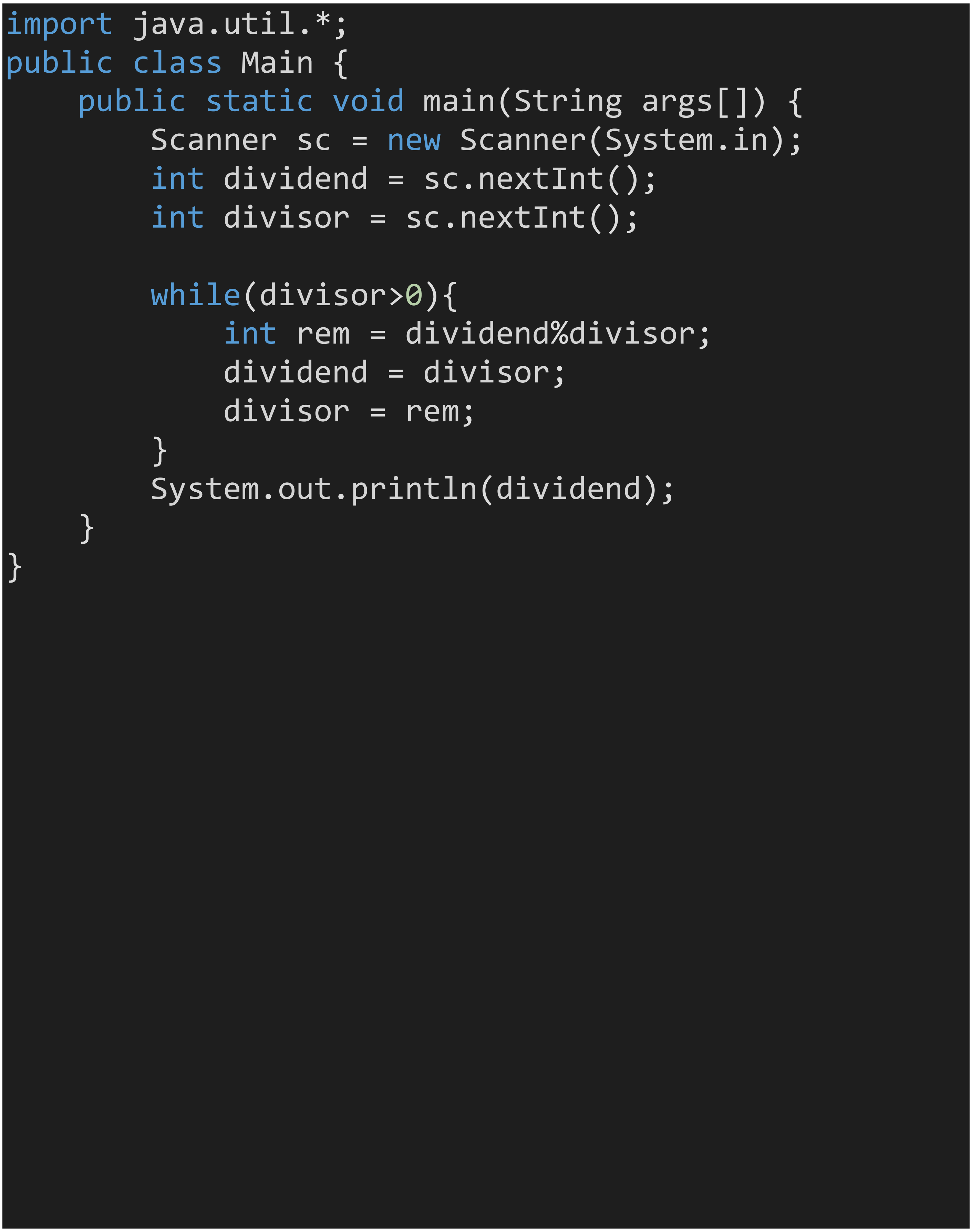
```
16
24
```

Sample Output

```
8
```

Explanation

The largest number that divides both N1 and N2 is called the GCD of N1 and N2.



```
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int dividend = sc.nextInt();
        int divisor = sc.nextInt();

        while(divisor>0){
            int rem = dividend%divisor;
            dividend = divisor;
            divisor = rem;
        }
        System.out.println(dividend);
    }
}
```

Take N (number of rows), print the following pattern (for N = 6)
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1

Input Format

Constraints

```
0 < N < 100
```

Output Format

Sample Input

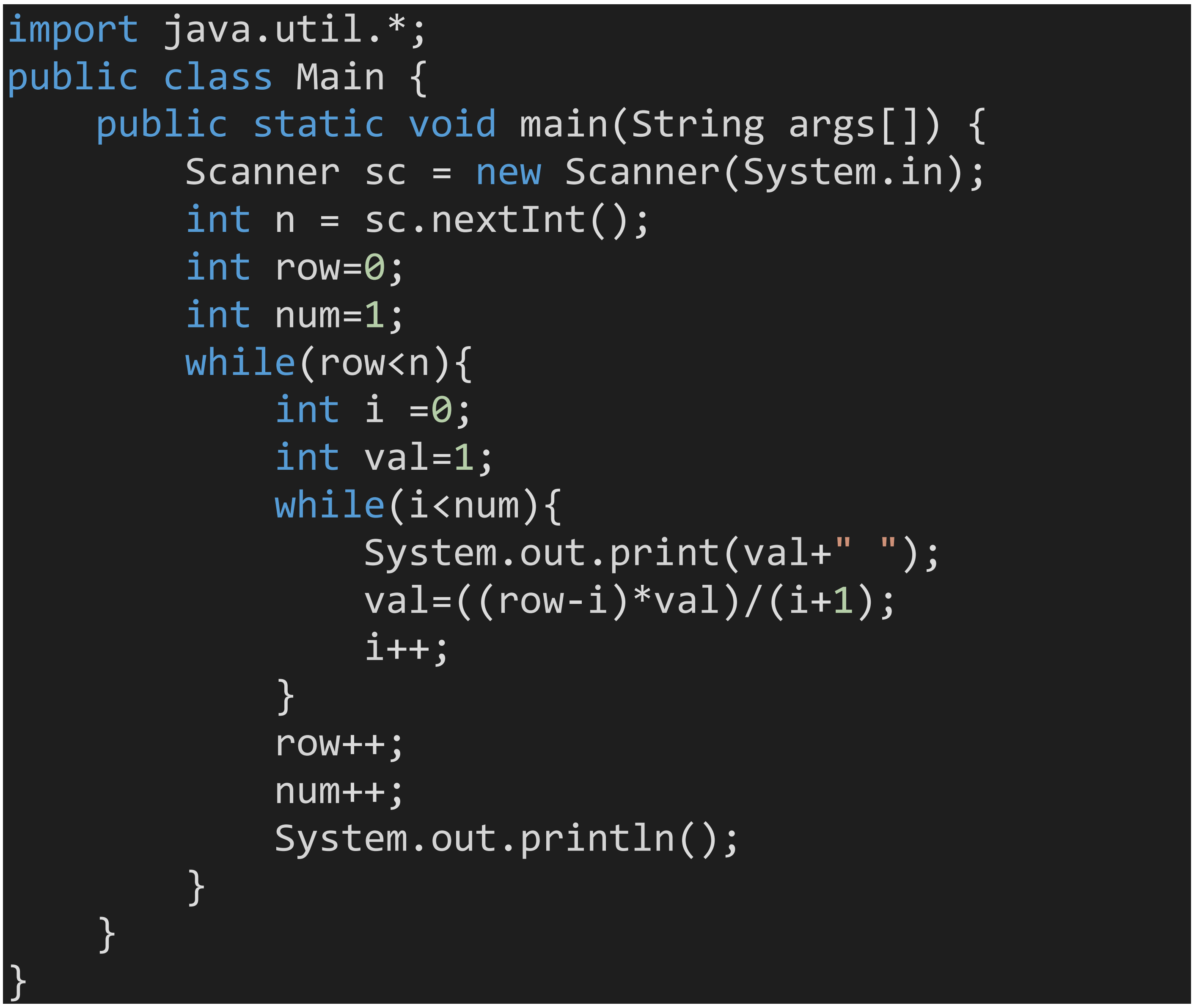
```
6
```

Sample Output

```
1
1       1
1       2       1
1       3       3       1
1       4       6       4       1
1       5       10      10      5       1
```

Explanation

```
Each number is separated from other by a tab.
```



```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int row=0;
        int num=1;
        while(row<n){
            int i =0;
            int val=1;
            while(i<num){
                System.out.print(val+"  ");
                val=((row-i)*val)/(i+1);
                i++;
            }
            row++;
            num++;
            System.out.println();
        }
    }
}
```

Take N (number of rows), print the following pattern (for N = 4).

```
1           1
1 2       2 1
1 2 3   3 2 1
1 2 3 4 3 2 1
```

Input Format

Constraints

```
0 < N < 10
```

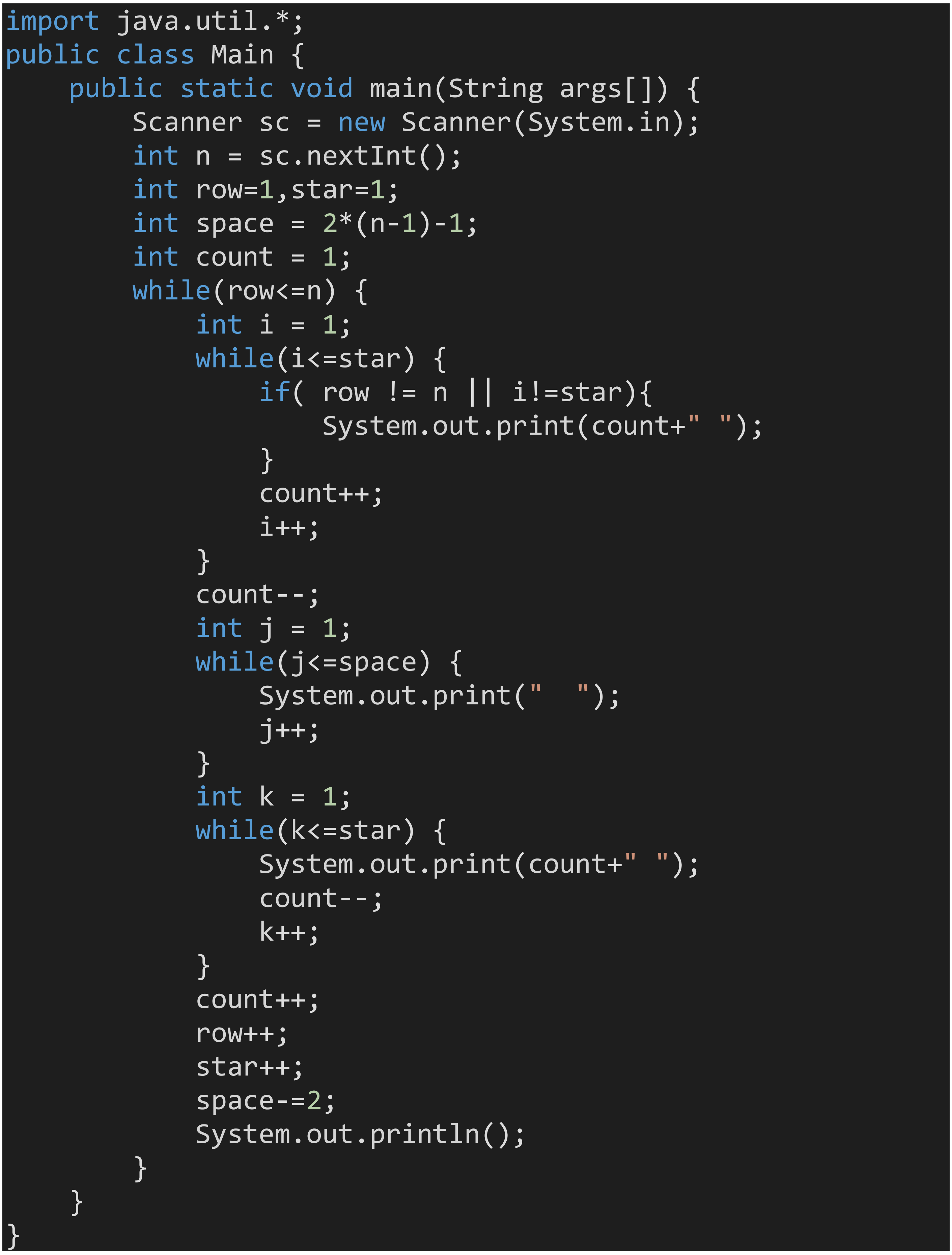
Output Format

Sample Input

```
4
```

Sample Output

```
1                                               1
1       2                               2       1
1       2       3               3       2       1
1       2       3       4       3       2       1
```



```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int row=1,star=1;
        int space = 2*(n-1)-1;
        int count = 1;
        while(row<=n) {
            int i = 1;
            while(i<=star) {
                if( row != n || i!=star){
                    System.out.print(count+" ");
                }
                count++;
                i++;
            }
            count--;
            int j = 1;
            while(j<=space) {
                System.out.print("  ");
                j++;
            }
            int k = 1;
            while(k<=star) {
                System.out.print(count+" ");
                count--;
                k++;
            }
            count++;
            row++;
            star++;
            space-=2;
            System.out.println();
        }
    }
}
```

Faculty at CodingBlocks loves to purchase smartphones and decides to play a game. Aayush and Harshit decides to shop for smartphones. Aayush purchases 1 smartphone, then Harshit purchases 2 smartphones, then Aayush purchases 3 smartphones, then Harshit purchases 4 smartphones, and so on. Once someone can't purchase more smartphones, he loses.

Aayush can purchase at most M smartphones and Harshit can purchase at most N smartphones. Who will win ? Print "Aayush" and "Harshit" accordingly.

Input Format

The first line of the input contains an integer T denoting the number of test cases. The description of T test cases follows. Two integers M and N denoting the maximum possible number of smartphones Aayush and Harshit can purchase respectively.

Constraints

```
1 ≤ T ≤ 1000 1 ≤ M, N ≤ 10^6
```

Output Format

For each test case, output a single line containing one string — the name of the winner i.e. Aayush or Harshit

Sample Input

```
2
9 3
8 11
```

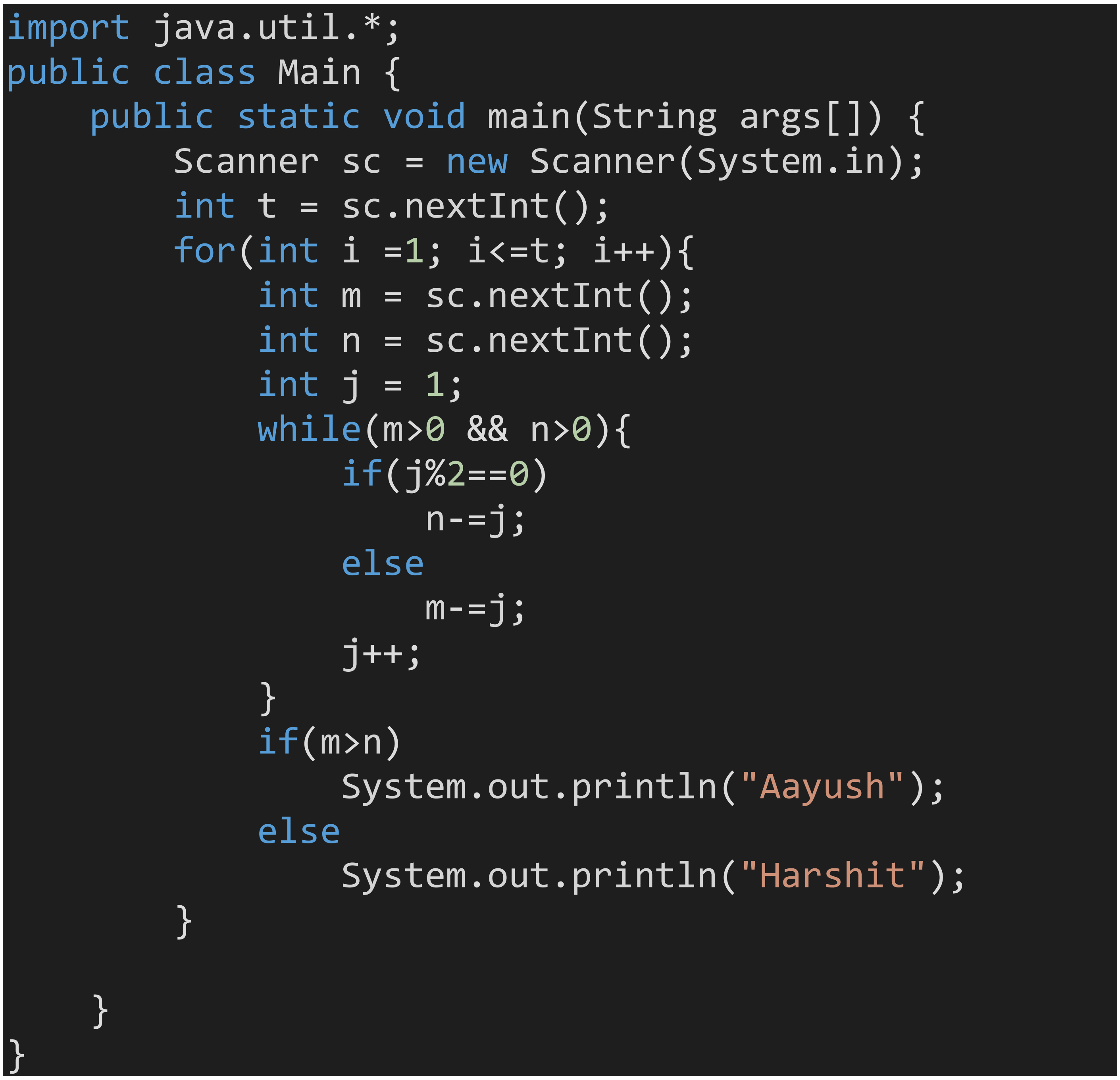
Sample Output

```
Aayush
Harshit
```

Explanation

Test case 1. We have M = 9 and N = 3. Aayush shops for 1 smartphone, and then Harshit shops for 2 smartphones. Then Aayush shops for 3 smartphones and then Harshit shops for 4 smartphones but that would mean 2 + 4 = 6 smartphones in total. It's impossible because Harshit can shop for at most N smartphones, so he loses. Aayush wins, and so we print "Aayush".

Test case 2. Now we have M = 8 and N = 11. Aayush shops for 1 smartphone first, and then Harshit shops for 2 smartphones, then Aayush shops for 3 smartphones. Now Harshit shops for 4 smartphones (he has 2 + 4 = 6 smartphones in total, which is allowed because it doesn't exceed N). Then Aayush tries to shop for 5 smartphones but he can't because 1 + 3 + 5 = 9 which is greater than M. Aayush loses and Harshit is the winner.



```java
import java.util.*;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for(int i =1; i<=t; i++){
            int m = sc.nextInt();
            int n = sc.nextInt();
            int j = 1;
            while(m>0 && n>0){
                if(j%2==0)
                    n-=j;
                else
                    m-=j;
                j++;
            }
            if(m>n)
                System.out.println("Aayush");
            else
                System.out.println("Harshit");
        }

    }
}
```