while.

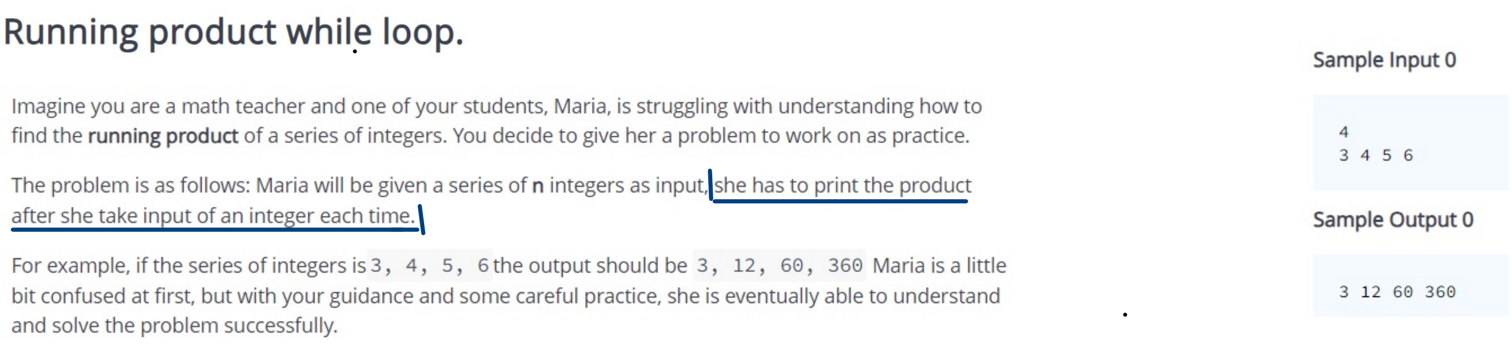
running sum → for loop.

3

2 7 1

o/p. 2 9 10

*



Running product while loop.

Imagine you are a math teacher and one of your students, Maria, is struggling with understanding how to find the **running product** of a series of integers. You decide to give her a problem to work on as practice.

The problem is as follows: Maria will be given a series of **n** integers as input, she has to print the product after she take input of an integer each time.

For example, if the series of integers is `3, 4, 5, 6` the output should be `3, 12, 60, 360` Maria is a little bit confused at first, but with your guidance and some careful practice, she is eventually able to understand and solve the problem successfully.

Sample Input 0

```
4
3 4 5 6
```

Sample Output 0

```
3 12 60 360
```

n → 4

3 4 5 6

↑

ans = ~~1~~ ~~3~~ ~~12~~ ~~60~~ 360

ans * = x

algo

ntimes.

- i/p
- find product.
- print product.

3 12 60 360

ans = 1

3

2 7 1

o/p ↳ 2 9 10

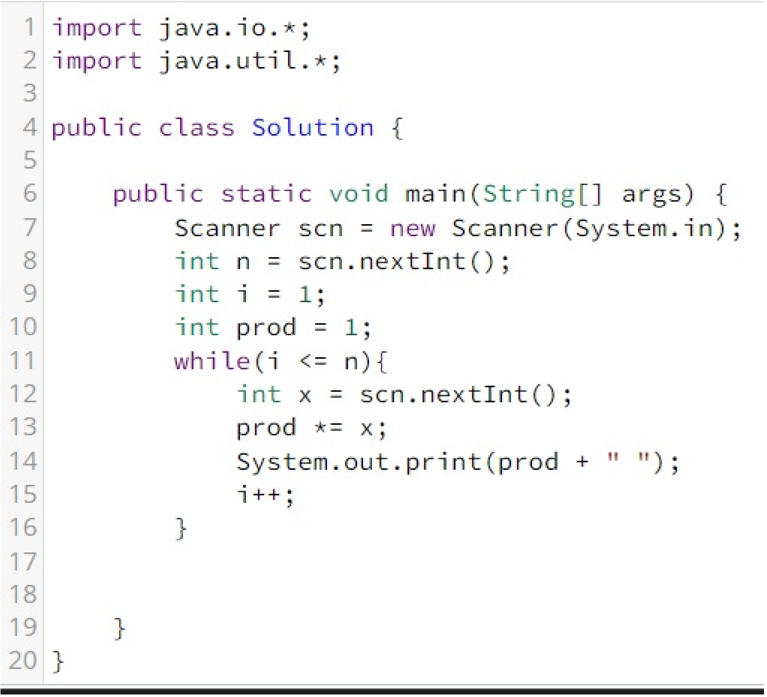
n = 4 ✓

3 4 5 6

↑

prod = ~~1~~ ~~3~~ ~~12~~ ~~60~~ 360

3 12 60 360



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int i = 1;
        int prod = 1;
        while(i <= n){
            int x = scn.nextInt();
            prod *= x;
            System.out.print(prod + " ");
            i++;
        }


    }
}
```

n = 3

3 4 2

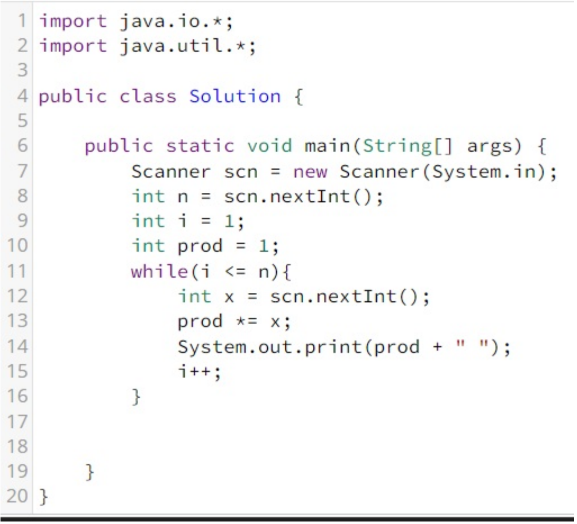
↑

prod = ~~1~~ ~~3~~ ~~12~~ 24

n times

- → i/p ✓
- → prod ✓
- → print

3 12 24



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int i = 1;
        int prod = 1;
        while(i <= n){
            int x = scn.nextInt();
            prod *= x;
            System.out.print(prod + " ");
            i++;
        }


    }
}
```

<u>3</u>

3 4 2

↑

n =3

prod = ~~1~~ ~~3~~ ~~12~~ 24

i = ~~1~~

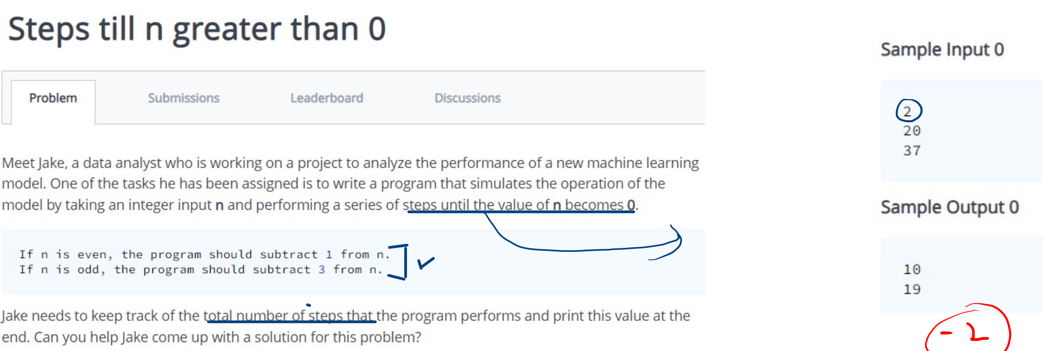
1 ≤ 3 ✓ x = 3

~~2~~ 2 ≤ 3 x = 4

~~3~~ 3 ≤ 3 ✓ x = 2

4 ≤ 3 ✗

3 ..... 12 ..... 24 ...



Steps till n greater than 0

Meet Jake, a data analyst who is working on a project to analyze the performance of a new machine learning model. One of the tasks he has been assigned is to write a program that simulates the operation of the model by taking an integer input **n** and performing a series of steps until the value of **n** becomes **0**.

```
If n is even, the program should subtract 1 from n.
If n is odd, the program should subtract 3 from n.
```

✓

Jake needs to keep track of the total number of steps that the program performs and print this value at the end. Can you help Jake come up with a solution for this problem?

Sample Input 0

```
2
20
37
```

Sample Output 0

```
10
19
```

2 → 2 test case.

20 → TC1

37 → TC2

TC1

n = 20 ↓ 19 ↓ 16 ↓ 15 ↓ 12 ↓ 11 ↓ 8 ↓ 7 → 4 → 3 → 0

steps = ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ 10

-2

TC2 = ~~37~~ ~~1~~

34

33

30

29

26

25

22

~~21~~

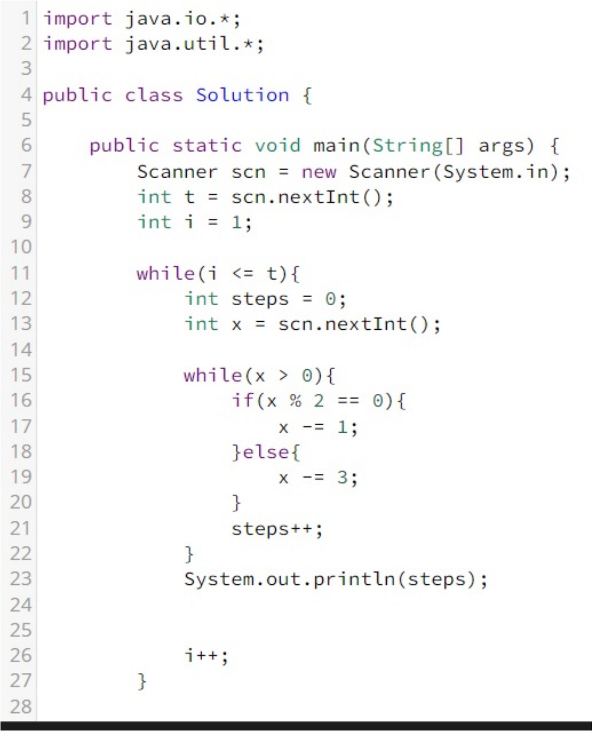
~~18~~

17 — 14 | 13 | 10 | 9 | ~~6~~ ~~5~~ ~~2~~

steps = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~ ~~11~~ ~~12~~ ~~13~~ ~~14~~ ~~15~~ ~~16~~ ~~17~~ ~~18~~ 19

1

n = 3



```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int t = scn.nextInt();
        int i = 1;

        while(i <= t){
            int steps = 0;
            int x = scn.nextInt();

            while(x > 0){
                if(x % 2 == 0){
                    x -= 1;
                }else{
                    x -= 3;
                }
                steps++;
            }
            System.out.println(steps);


            i++;
        }

```

o/p

3

4

t = 2

$n_1 \to 5 \to$ ✓

$n_1 \to 8 \to$ ✓

t = 2

i = ~~1~~

~~2~~

3

$1 \le 2$ ✓

$2 \le 2$ ✓

$3 \le 2$ ✗

steps = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4

x = ~~8~~ ~~7~~ ~~4~~ ~~3~~ 0

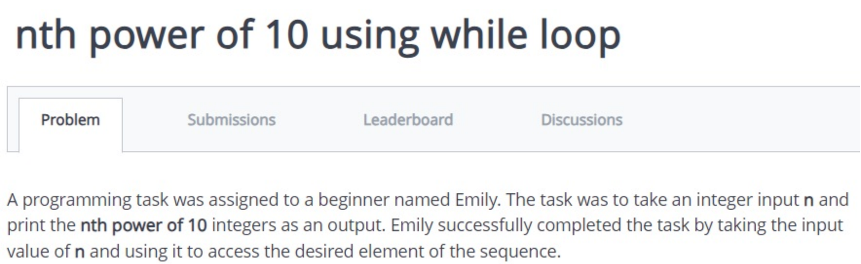
8 > 0

7 > 0

4 > 0

3 > 0

0 > 0 ✗



nth power of 10 using while loop

A programming task was assigned to a beginner named Emily. The task was to take an integer input **n** and print the **nth power of 10** integers as an output. Emily successfully completed the task by taking the input value of **n** and using it to access the desired element of the sequence.

$n = 0$

$10^0 = \boxed{1}$

$n = 2$

$10^2 = \boxed{100}$

$n = 5.$

$10^5 = \boxed{100000}$

$\boxed{n = 3} \rightarrow 1000$

$\downarrow$

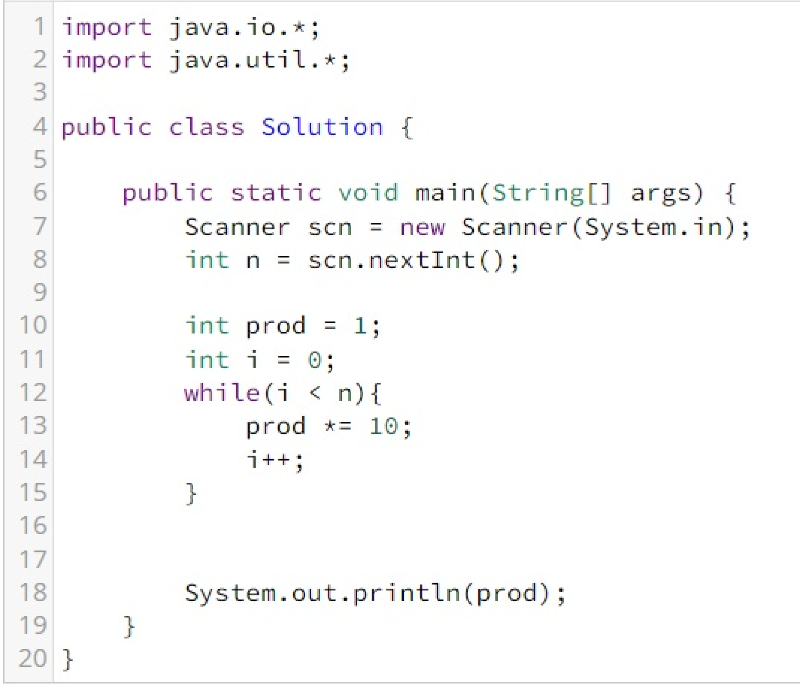
$1 \times \underbrace{10 \times 10 \times 10}$

$prod = 1$

$n = 4 \longrightarrow 10000$

$\downarrow$

$1 \times \; 10 \times 10 \times 10 \times 10$



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();

        int prod = 1;
        int i = 0;
        while(i < n){
            prod *= 10;
            i++;
        }


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

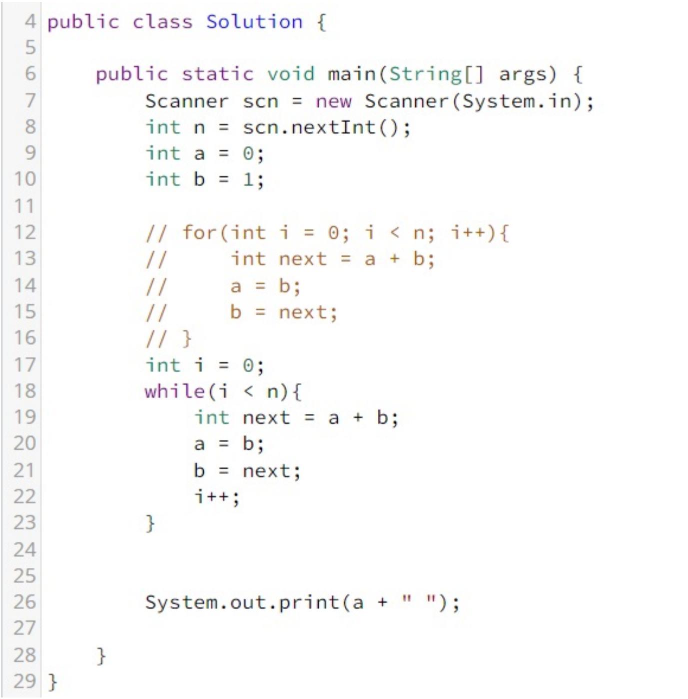
$n=0$

$prod=1$ ✓

$i=0$

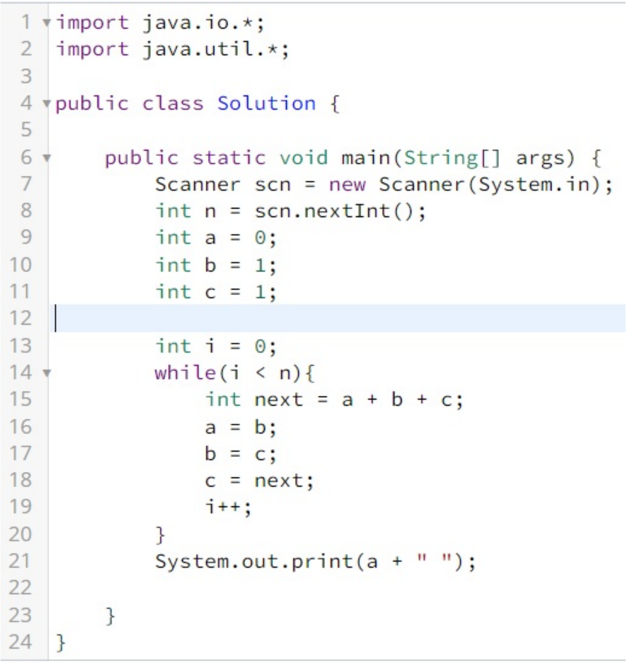
$0<0$ f

fibo with while loop.



```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int a = 0;
        int b = 1;

        // for(int i = 0; i < n; i++){
        //     int next = a + b;
        //     a = b;
        //     b = next;
        // }
        int i = 0;
        while(i < n){
            int next = a + b;
            a = b;
            b = next;
            i++;
        }


        System.out.print(a + " ");

    }
}
```

tribo with while loop.

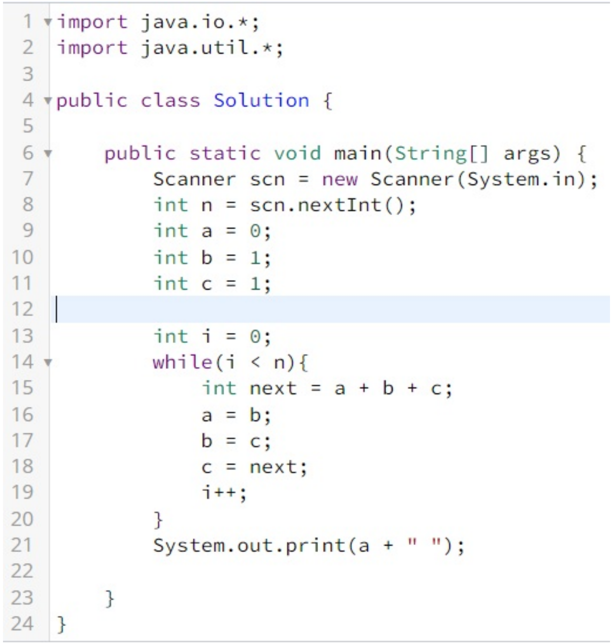


```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int a = 0;
        int b = 1;
        int c = 1;

        int i = 0;
        while(i < n){
            int next = a + b + c;
            a = b;
            b = c;
            c = next;
            i++;
        }
        System.out.print(a + " ");

    }
}
```

Print nth Tribonacci number

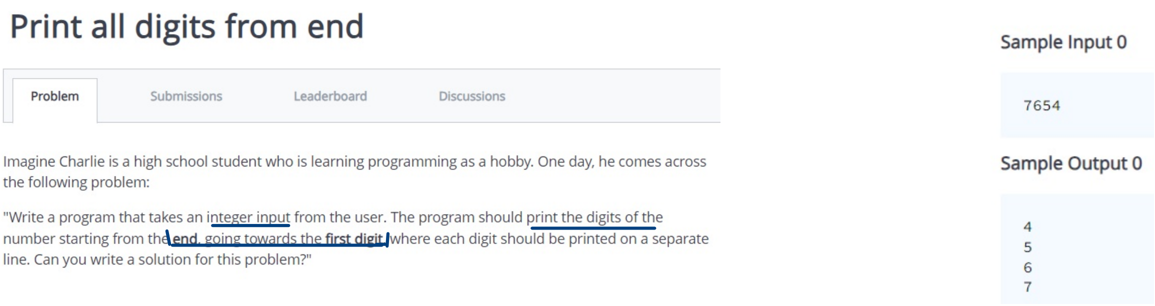
0	1	2	3	4	5	6	7	
0	1	1	2	4	7	13	24	44 ...

$n=7 \longrightarrow 24$



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int a = 0;
        int b = 1;
        int c = 1;

        int i = 0;
        while(i < n){
            int next = a + b + c;
            a = b;
            b = c;
            c = next;
            i++;
        }
        System.out.print(a + " ");

    }
}
```

d



Print all digits from end

Imagine Charlie is a high school student who is learning programming as a hobby. One day, he comes across the following problem:

"Write a program that takes an integer input from the user. The program should print the digits of the number starting from the **end, going towards the first digit** where each digit should be printed on a separate line. Can you write a solution for this problem?"

Sample Input 0

```
7654
```

Sample Output 0

```
4
5
6
7
```

eg n = 7654

4
5
6
7

repeat

7654
ld
reduce number (765)

n

ld = 7654 % 10 = 4 ✓

765 % 10 = 5

76 % 10 = 6

7 % 10 = 7

7654/10 → 765 ?

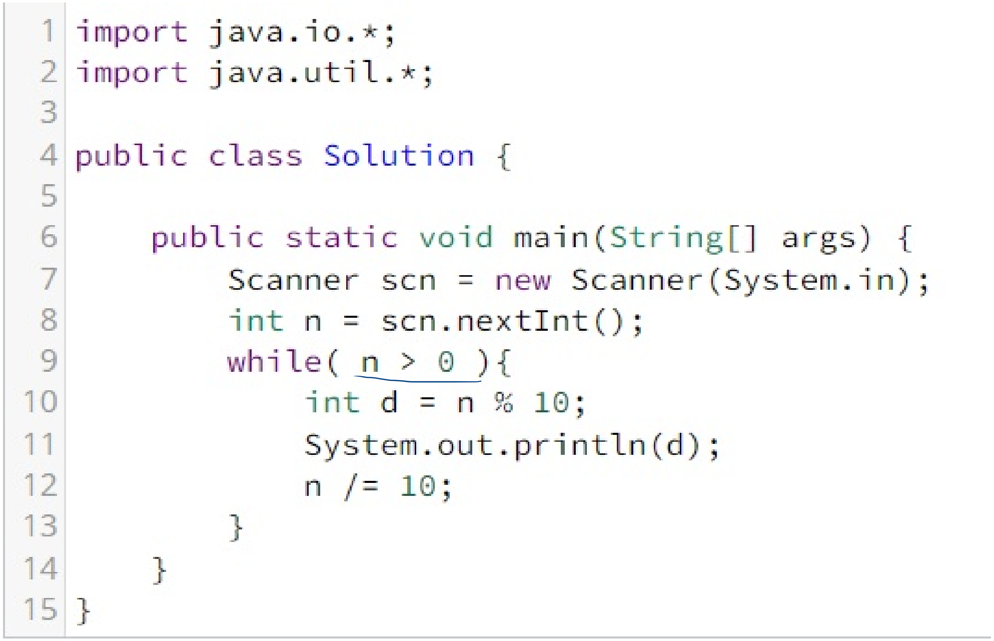
$$10\,\overline{)\,7654\,}\,(\,765$$

7650

4

~~4~~
~~3~~
2
1

n = 1 2 3 4



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        while( n > 0 ){
            int d = n % 10;
            System.out.println(d);
            n /= 10;
        }
    }
}
```

1234 > 0

d = 4

↓

123

d = 3

↓

12

GKSTR46 Number of Digits

logic

Count $= \cancel{0}\ \cancel{1}\ \cancel{2}\ 3$

eg.

$n = 523 \rightarrow \boxed{3}$

$n = 1234 \rightarrow \boxed{4}$

523

↓

52

↓

5

↓

0

count $= \cancel{0}\ \cancel{1}\ \cancel{2}\ \cancel{3}\ \underline{4}$

$n = 1234$

↓

123

↓

12

↓

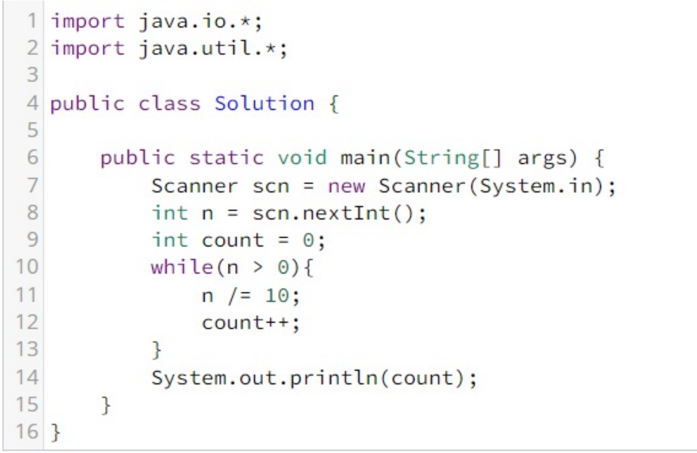
1

↓

0

n = 1234

count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4



```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int count = 0;
        while(n > 0){
            n /= 10;
            count++;
        }
        System.out.println(count);
    }
}
```

1234 > 0

↓

123 > 0

↓

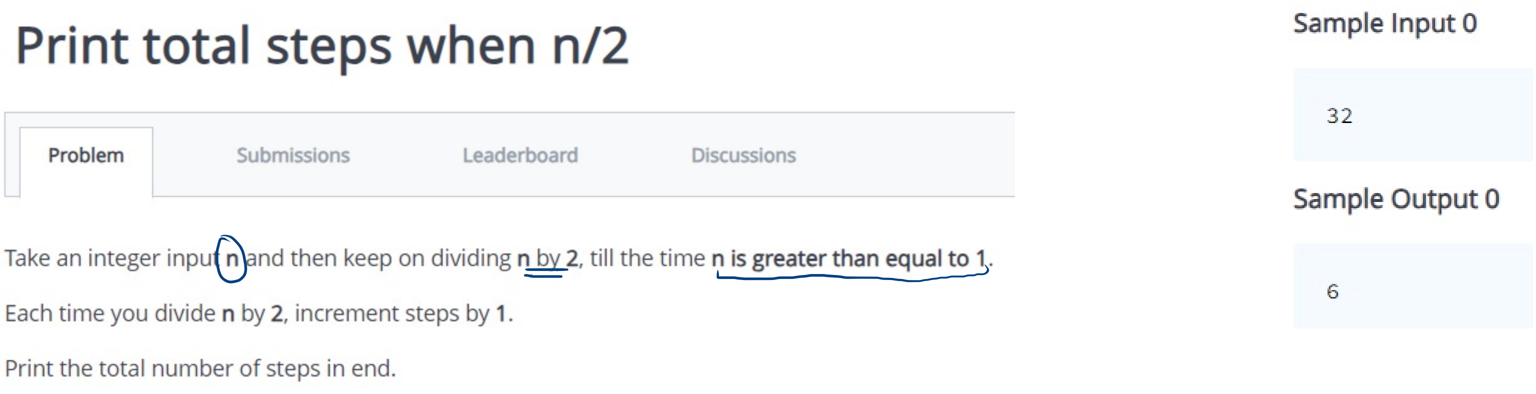
12 > 0

↓

1 > 0

↓

0 > 0 ✗



Print total steps when n/2

Take an integer input n and then keep on dividing n by 2, till the time n is greater than equal to 1.

Each time you divide **n** by **2**, increment steps by **1**.

Print the total number of steps in end.

Sample Input 0

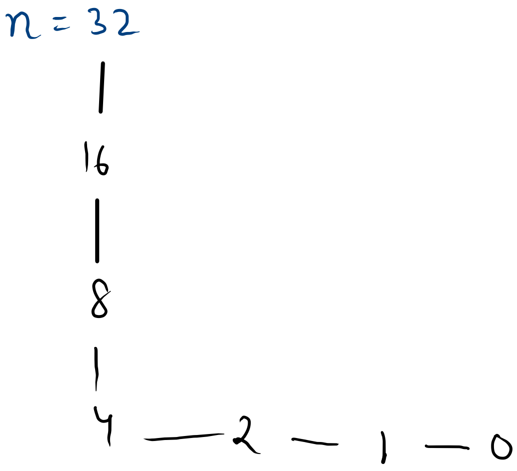
```
32
```

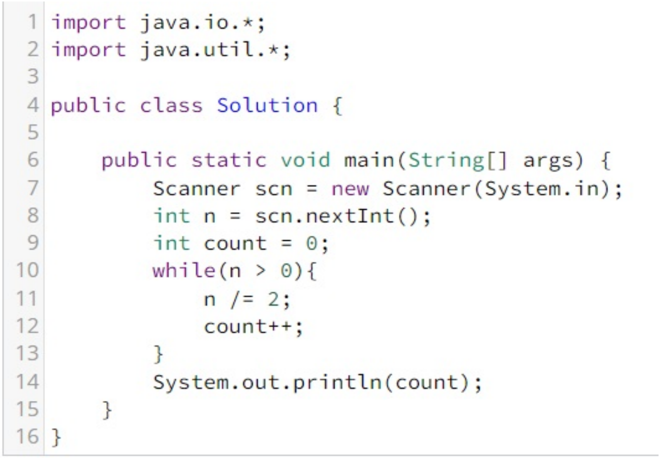
Sample Output 0

```
6
```

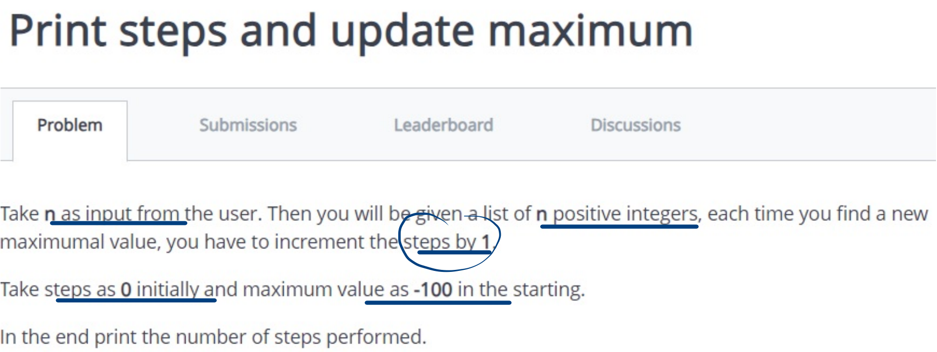
$n \geq 1$

Steps = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6





```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int count = 0;
        while(n > 0){
            n /= 2;
            count++;
        }
        System.out.println(count);
    }
}
```



Print steps and update maximum

Take **n** as input from the user. Then you will be given a list of **n** positive integers, each time you find a new maximumal value, you have to increment the steps by **1**.

Take steps as **0** initially and maximum value as **-100** in the starting.

In the end print the number of steps performed.

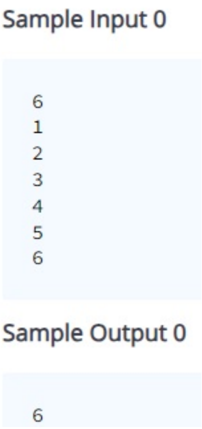
steps = ~~0~~ ~~1~~ ~~2~~ 3

max = ~~-100~~ ~~2~~ ~~7~~ 8

```
if (x > max)
{
    max = x
    steps++
}
```

eg 5

2
1
7
8
3



Sample Input 0

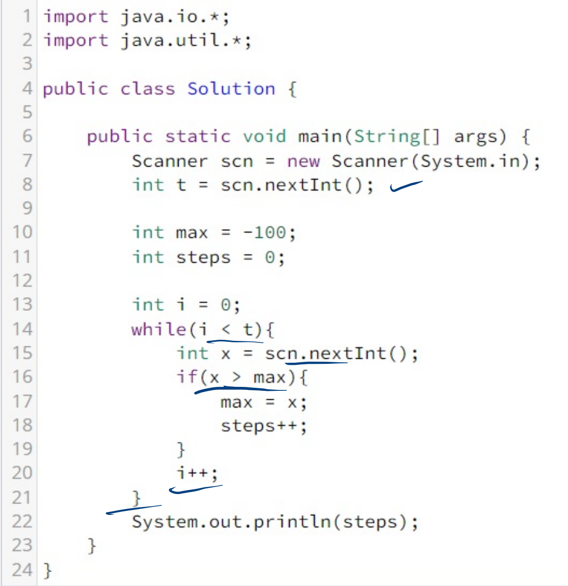
```
6
1
2
3
4
5
6
```

Sample Output 0

```
6
```

4

1
2
3
4



```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int t = scn.nextInt();

        int max = -100;
        int steps = 0;

        int i = 0;
        while(i < t){
            int x = scn.nextInt();
            if(x > max){
                max = x;
                steps++;
            }
            i++;
        }
        System.out.println(steps);
    }
}
```

t=5

max = ~~-100~~ ~~2~~ ~~6~~ 7

steps = ~~0~~ ~~1~~ ~~2~~ 3

5

2 ✓
6
4 ✓
7 ✓
3 ✓

i = ~~0~~ 0<5
x=2
2>-100

= ~~1~~ 1<5
x=6
6>2

~~2~~ 2<5
x=4
4>6

~~3~~ 3<5
x=7
7>6

~~4~~ 4<5
x=3
3>7

5 5<5