Running product while loop.

→ product of all the elements on left side including itself

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

    int product = 1;
    int i = 0;
    while (i < n) { // n times
        int val = scn.nextInt();
        product = product * val;
        System.out.print(product + " ");
        i++;
    }
}
```

Steps till n greater than 0 (do what it says)

n = 20

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

```
n = 20; ← input
while ( n > 0 ) {
    if ( n is even ) {
        n--;
    } else {
        n -= 3;
    }
}
```

input

T = 3;

n = 20,

n = 10,

n = 15,

dry run

T = 2

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

    for (int i = 0; i < T; i++) { // T times
        int n = scn.nextInt();
        int steps = 0;
        while ( n > 0 ) {
            if ( n % 2 == 0 ) {
                n -= 1;
            } else {
                n -= 3;
            }
            steps++;
        }
        System.out.println(steps);
    }
}
```

i=0, n=10

steps = ~~0~~ 1 2 3 4 5 6,

(10 > 0) ✓

(9 > 0) ✓

(6 > 0) ✓

(5 > 0) ✓

(2 > 0) ✓

(1 > 0) ✓

(-2 > 0) ✗

i=1, n=3,

o/p :- 6

nth power of 10 using while loop

$n = 4$

$ans = 10^4 = 10 * 10 * 10 * 10$

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

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

    int ans = 1;
    int i = 0;
    while (i < n) {
        ans = ans * 10;
        i++;
    }
    System.out.println(ans);
}
```

n = 4;

ans = 1 * 10 * 10 * 10 * 10

i = 0, (0 < 4) ✓

i = 1, (1 < 4) ✓

i = 2, (2 < 4) ✓

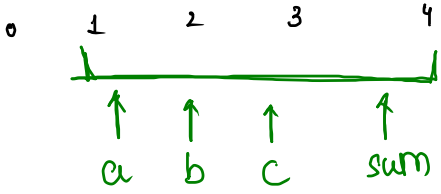
i = 3, (3 < 4) ✓

i = 4, (4 < 4) X

ans = 10000

Print nth Tribonacci number

series:- 0, 1, 1, 2, 4, 7, 13, ------



Code

```
if (n == 0){
    print 0
} else if (n == 1){
    print 1
} else if (n == 2){
    print 1
} else {
    int a = 0
    int b = 1
    int c = 1
    int sum = 0
    loop (n-2) times
        sum = a+b+c,
        a = b
        b = c
        c = sum
}
```

Code

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

    if (n == 0) {
        System.out.println(0);
    } else if (n == 1) {
        System.out.println(1);
    } else if (n == 2) {
        System.out.println(1);
    } else {

        int a = 0;
        int b = 1;
        int c = 1;
        int sum = 0;
        for (int i = 3; i <= n; i++) {
            sum = a + b + c;
            a = b;
            b = c;
            c = sum;
        }
        System.out.println(sum);
    }
}
```

Print all digits from end

print 5 4 3 2 1

n = 12345;

rem = n%10 , n = n/10 , n > 0

rem = 12345%10 = 5 , n = 12345/10 = 1234 , 1234 > 0

rem = 1234%10 = 4 , n = 1234/10 = 123 , 123 > 0

rem = 123%10 = 3 , n = 123/10 = 12 , 12 > 0

rem = 12%10 = 2 , n = 12/10 = 1 , 1 > 0

rem = 1%10 = 1 , n = 1/10 = 0 , 0 > 0 X

Code

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

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

GKSTR46 Number of Digits

n = 42035; count = 0

(42035 > 0) , n = 42035/10 , count = 1
= 4203

(4203 > 0) ✓ , n = 4203/10 , count = 2
= 420

(420 > 0) ✓ , n = 420/10 , count = 3
= 42

(42 > 0) ✓ , n = 42/10 , count = 4
= 4

(4 > 0) ✓ , n = 4/10 , count = 5
= 0

Code

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

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**.

one liner:- from n to n>=1 by n=n/2

```
for(int i=n; i>=1; i=i/2){
    steps++;
}
```

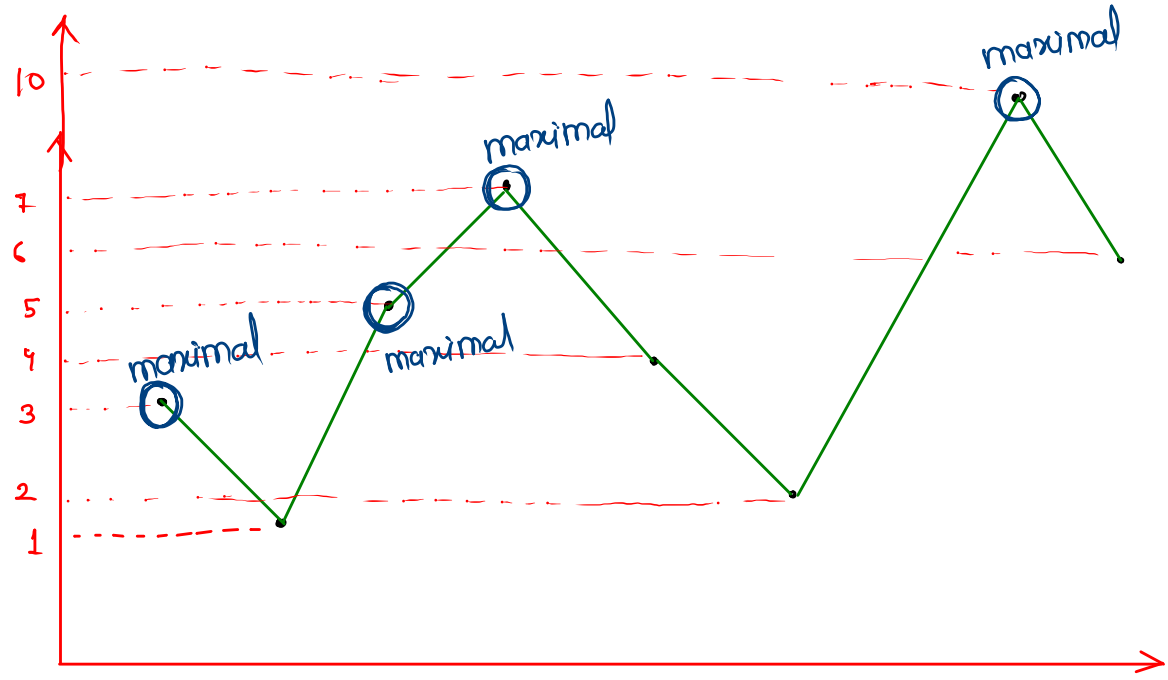
```
int i=n;
while(i>=1){
    steps++;
    i=i/2;
}
```

Code

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

    int steps = 0;
    int i = n;
    while (i >= 1) {
        steps++;
        i /= 2;
    }
    System.out.println(steps);
}
```

Print steps and update maximum



maximum value :- largest value

maximal value :- largest value till now

Note:- all maximum values are maximal but vice versa is not true

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

    int count = 0;
    int maximal = -100;
    for (int i = 0; i < n; i++) {
        int val = scn.nextInt();
        if ( val > maximal ) {
            count++;
            maximal = val;
        }
    }
    System.out.println(count);
}
```

n = 5 ;

count = ~~0 1 2~~ 3

maximal = ~~-100 2 4~~ 7

i = 0, val = 2, (2 > -100) ✓

i = 1, val = 4, (4 > 2) ✓

i = 2, val = 3, (3 > 4) ✗

i = 3, val = 7, (7 > 4) ✓

i = 4, val = -2, (-2 > 7) ✗

graph