Today's agenda

- No. of factors
- Prime numbers
- Sum of N natural no.s
- floor & ceil
- Sqrt
- ....

Q) Count no. of factors

↳ Given a number N, Print the count of factors.

N = 24 → {1 2 3 4 6 8 12 24} → 8

N = 36 → {1 2 3 4 6 9 12 18 36} → 9

```
P S V main () {
    Scanner scn = new Scanner (System.in);
    int n = scn.nextInt();
    int count = 0;
    for (int i = 1; i <= N; i++) {
        if (N % i == 0) {
            count ++;
        }
    }
}
```

iteration count

N ↓

N = 10^{18} → Seconds?

↳ No. of iteration = 10^{18}

10^8 iterations = 1 sec

1 iteration = $\frac{1}{10^8}$ sec

10^{18} iteration : $\frac{1}{10^8} \times 10^{18} = 10^{10}$ sec

10^{10} sec = 317 years → 10 sec

you → child → grandchild → 3rd → 4th → 5th gen

By forces

1 sec = 10^8 iterations

No. of iterations = N

N = 10^9

10^9 iterations

10^8 iterations = 1 sec

1 iteration = $\frac{1}{10^8}$ sec

$1 * 10^9 = \frac{1}{10^8} * 10^9$ sec

= 10 sec

//optimize

$i * j = N \Rightarrow j = N/i$

$\Downarrow$

i & N/i are the factors

N = 24

i		j = N/i	
1	<	24	+2
2	<	12	+2
3	<	8	+2
4	<	6	+2
6	>	4	
8	>	3	
12	>	2	
24	>	1	

$i <= N/i$

$i^2 <= N$

$i <= \sqrt{N}$

N = 36

i		N/i	
1	<	36	+2
2	<	18	+2
3	<	12	+2
4	<	9	+2
6	==	6	+1
9		4	
12		3	
18		2	
36		1	

// Pseudo code

```
int countFactors (int N){
    int count = 0;
    for (int i=1; i*i<=N; i++){
        if (N % i == 0){
            if (i == N/i) { count = count + 1; }
            else { count = count + 2; }
        }
    }
    return count;
}
```

No. of iteration ∝ $\sqrt{N}$ iteration

N = 24

```
int count = 0;
for (int i=1; i*i<N; i++){
    if (N % i == 0){
        count = count + 2;
    }
}
```

N = 24

count = 0

i	i*i<N	N%i	N/i	count
1	+	+		2
2	+	+		4
3	+	+		6
4	+	+		8
5	b			

b → exit

N = 36

```
int Count = 0;
for (int i = 1; i*i <= N; i++) {
    if (N % i == 0) {
        if (i == N/i) { Count = Count + 1; }
        else { Count = Count + 2; }
    }
}
```

Count = 0

i	i*i <= N	N % i	i == N/i	N/i	Count
1	+	+	F		2 → 34
2	+	+	F	4 → 18	
3	+	+	F	6 → 12	
4	+	+	F	8 → 9	
5	+	F			
6	F	+	+	9 → 6	
7	F				

↳ exit

$\sqrt{N}$ iterations

↓

$N = 10^{18}$

↓

No. of iterations = $\sqrt{10^{18}} = 10^9$

10^8 iterations = 1 sec

1 iteration = $\frac{1}{10^8}$ sec

10^9 iterations : $\frac{1}{10^8} \times 10^9$

= 10 secs

Q) Prime numbers

↳ Given a number N, check if the number is a Prime no.

Prime numbers ⇒ Count of factors == 2

```
boolean isPrime (int N){
    int Count = 0;
    for (int i=1; i*i<=N; i++){
        if (N%i == 0){
            if (i==N/i) { Count = Count + 1; }
            else { Count = Count + 2; }
        }
    }
    if (Count == 2){ return true; }
    else { return false; }
}
```

→ true ⇒ N is Prime
↳ false ⇒ N is non-Prime

No. of iteration
↓
$\sqrt{N}$ iteration

Quiz 1: Sum of all the numbers from 1 to 10.

↳ $\frac{10 * 11}{2} = \frac{110}{2} = 55$

Quiz 2: Sum of all numbers from 1 to 1000.

↳ $\frac{1000 * 1001}{2} = 500500$

Q) Sum of first N natural numbers.

→ Gauss (4th class)

$S = 1 + 2 + 3 + 4 + \_ \_ \_ \_ + 998 + 999 + 1000$

$S = 1000 + 999 + 998 + 997 \_ \_ \_ \_ + 3 + 2 + 1$

$2*S = 1001 + 1001 + 1001 + 1001 + \_ \_ \_ \_ + 1001 + 1001 + 1001$

↓

$1001 * 1000$

$2 * S = 1001 * 1000$

$S = \frac{1001 * 1000}{2} = 500500$

// Sum of first N natural nos

$S: 1 + 2 + 3 \_ \_ \_ \_ + (N-2) + (N-1) + N$

$S: N + (N-1) + (N-2) \_ \_ \_ \_ + 3 + 2 + 1$

$(N+1) + (N+1) + (N+1) \quad + (N+1) + (N+1) + (N+1)$

$2*S = (N+1) * N \Rightarrow S = \frac{N * (N+1)}{2}$

Quiz3: Sum of 1^{st} N whole numbers.

↳ $0 + 1 + 2 + \_\_\_\_\_ + (N-1)$

$\Downarrow$

$1 + 2 + 3 + \_\_\_ + N-1 = \frac{(N-1) * (N-1+1)}{2}$

$= \frac{(N-1)^2 * N}{2}$

Sum of 1^{st} N whole numbers = Sum of first (N-1) natural numbers.

Break till 9:40pm

floor(num) → just smaller or equal integer

Ex: 7.4 → 7
8.9 → 8
100.01 → 100
90 → 90
20.99 → 20
3 → 3

Math.floor(num);

Ceil(num) → just greater or equal integer

Ex: 7.4 → 8

8.9 → 9

100.01 → 101

90 → 90

20.99 → 21

3 → 3

Math.ceil(num)

Q) Given N, return floor(sqrt(N))

ex: N = 60 → 7.... → 7
N = 31 → 5.x → 5
N = 29 → 5.y → 5
N = 16 → 4.0 → 4

floor($\sqrt{60}$) = 5 → 5*5 = 25
6*6 = 36
7*7 = 49
8*8 = 64 xx

```
int sqrt(int N){
    int ans=1;
    int i=1;
    while(i*i<=N){
        ans=i;
        i++;
    }
    return ans;
}
```

N = 60

i	i*i<=N	ans=i
1	t	1
2	t	2
3	t (4*4<=60)	3
4	t (5*5<=60)	4
5	t (6*6<=60)	5
6	t (7*7<=60)	6
7	t	7
8	f (exit)	

T.C. no. of iterations → $\sqrt{N}$

↳ you didn't come this far for only to come this far.