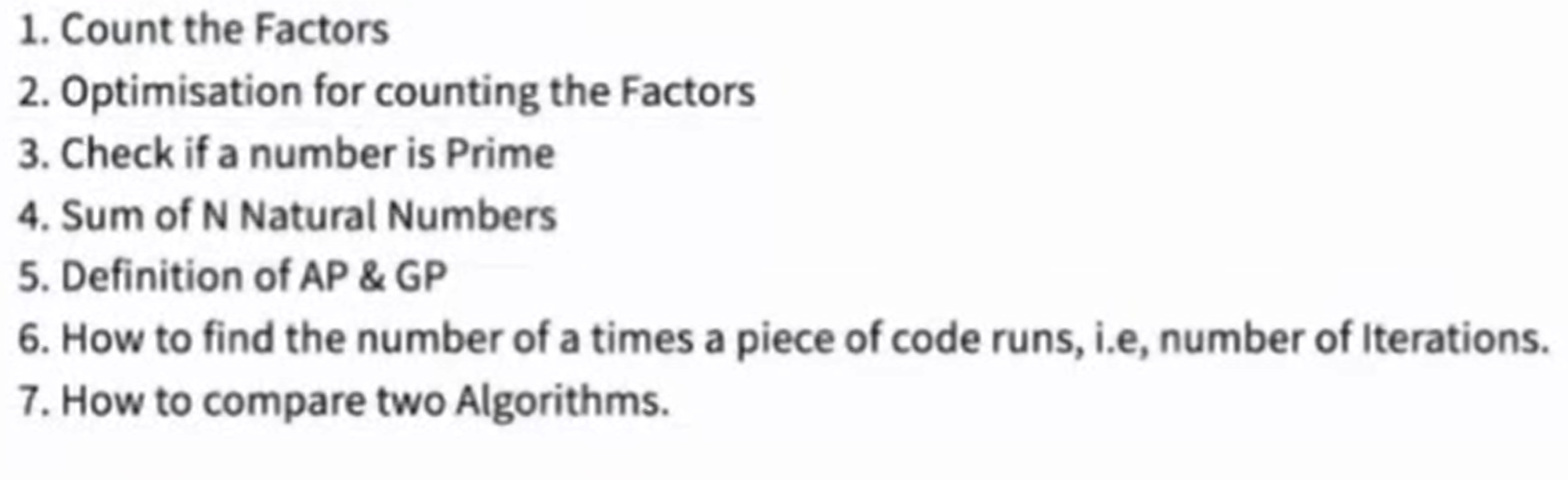


TODAYS CONTENT:



WHAT IS A FACTOR?

i is factor N when i is completely divide N

N%i == 0

Q. Given no. count factors of N. (N >0)

1. Number of factors of the number 24.

1, 2, 3, 4, 6, 7, 12, 24 🡪 8 factors

1. Number of factors of the number 10.

1, 2, 5, 10 🡪 4 factors

For N smallest factor is 1 and largest factor is N itself.

All the factors of N will be in the range of [1,N].

Pseudo code:

Function count\_factors(N) {

Count =0;

For i 🡪 1 to N{

If(N%i==0){

Count++;

}

}

Return count;

}

**Standard online compiler or code editor run 108 iterations in 1 sec.**

**108 Iteration 🡪 1 Sec.**

**1 Iteration 🡪 1/108 Sec.**

**N Iterations Time**

10 10 <1sec.

108 108 1 sec.

109 109 108\*10 = 1 sec\*10 = 10 sec

1018 1018 108\*1010 ≈ 317 Years

OPTIMIZATION:

i \* j = N {i & j are factors of N}

6 \* 3 = 18 {6 & 3 are factors of 18}

7 \* 8 = 56 {7 & 8 are factors of 56}

i \* j = N 🡪 j=N/i

N=24

i N/i

1 24

2 12

3 8

4 6

------------------------------------ After this point factors started repeating them.

6 4

8 3

12 2

24 1

By observing above we can say i<=N/i, after this factor are repeating

N=100

i N/i

1 100

2 50

4 25

5 20

10 10

----------------------------------- After this point factors started repeating them.

20 5

25 4

50 2

100 1

By observing above we can say i<=N/i, after this factor are repeating

i <= N/i

i2 <= N

i <=

OPTIMIZE CODE:

Function count\_factor(N) {

Count =0;

For (i 🡪 1 to : Math.Sqrt(N)) {

If (N%i ==0) {

If (i==N/i){

Count++;

} else {

Count +=2;

}

}

}

Return count;

}

N Iterations Time

1018 109 10 Sec.

Different way to write condition

1. i <=
2. i \* i <= N
3. i <= Math.sqrt(N) (Its internally does some more operation to calculate square root.

Q. Given N, check if its prime or not.

Prime number is number which having exact 2 factors.

Function isPrime(N){

If (count\_factors(N)==2){

Return True;

}else{

Return False;

}

}

1+2+3+4+5+………+100=5050

S = 1 + 2 + 3 + … + 100

S = 100 + 99 + 98 + … + 1

2S = 101 + 101 + 101 + … + 101

2S = 101 \* 100

S = (101 \* 100)/2

Generalize:

Sum of 1st N natural number.

S= 1 + 2 + 3 + … + (N-1) + N

S= N + (N-1) + (N-2) + … + 2 + 1

2S= N+1 + N+1 + N+1 + … + N+1 + N+1

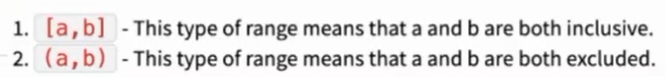
2S=(N+1) \* N

S=((N+1) \* N)/2

Some basic math properties:

[a,b] 🡪 all the numbers from a to b including a and b

(a,b) 🡪 all the numbers from a to b but exclude a and b.

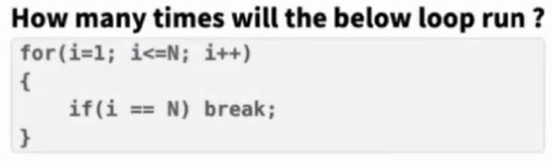


[2,4] 🡪 2,3,4

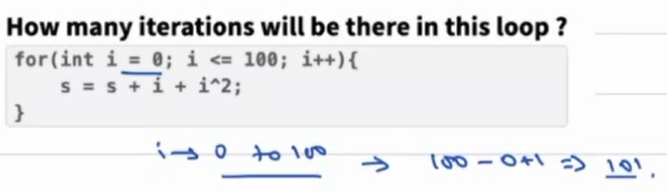
(2,4) 🡪 3

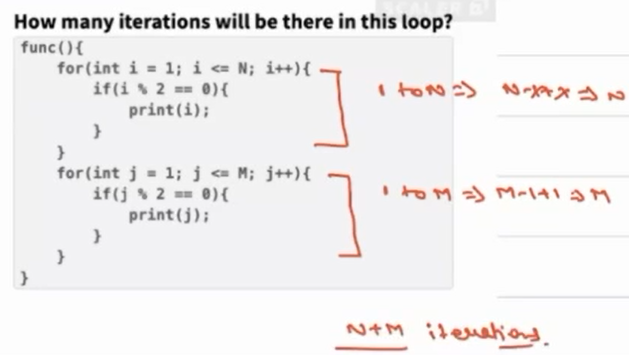
How many numbers are there in range [a,b]: b-a+1

ITERATION: The Number of time loop is run



ANS: N





Geometric Progression: (G.P)

The ratio of any 2 consecutive numbers will be same

5, 10, 20, 40, 80

10/5 20/10 40/20 80/20 =2 (Common ratio)

Generic Terminology:

ar0, ar1, ar2, ar3, ar4, …

If we are giving N terms with following G.P Pattern then sum is:

Sum🡪 a(rn-1)/(r-1) where r! = 1

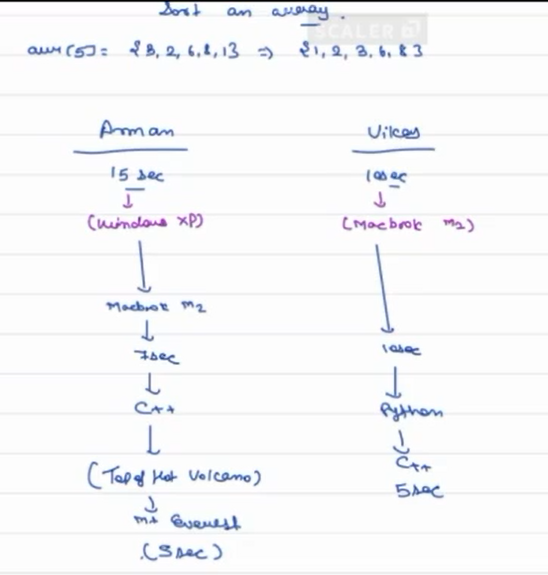
Sum🡪 a\*n where r=1

Arithmetic Progression:

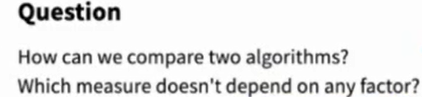
a, a+1d, a+2d, a+3d, …

sum🡪 n/2[2a+(n-1)d]

we cannot define algorithms performance on execution time because its depend on lot of factors like programming language used or system where we executing or place where we executing or OS etc.



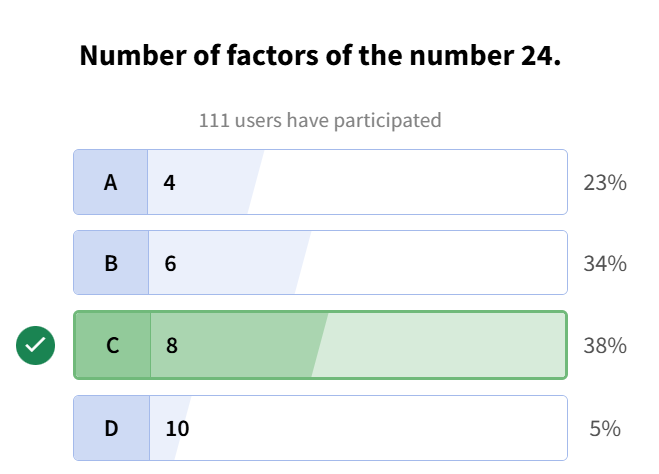


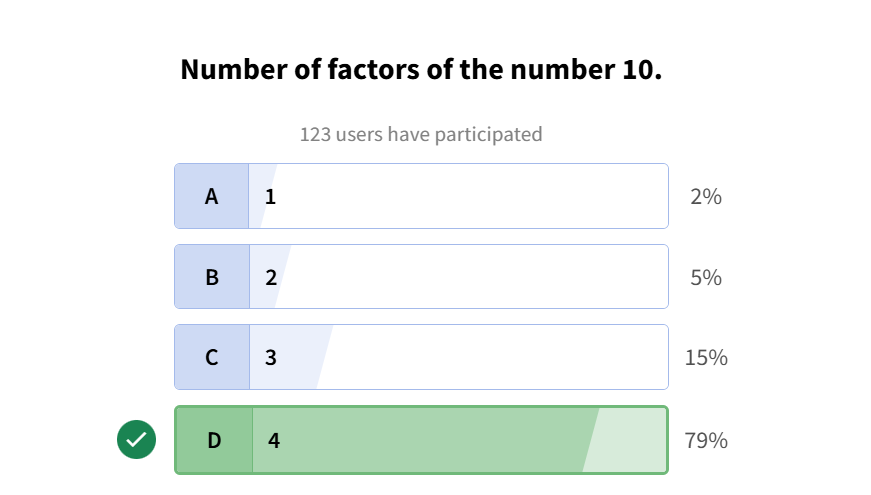


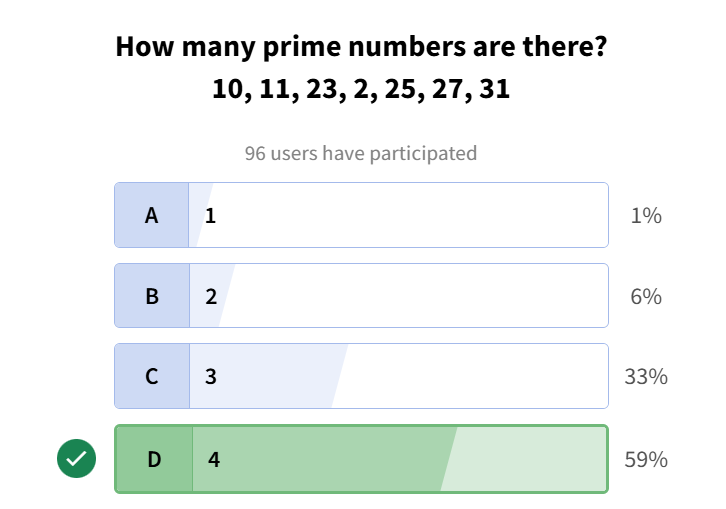
ANS: Number of iterations.

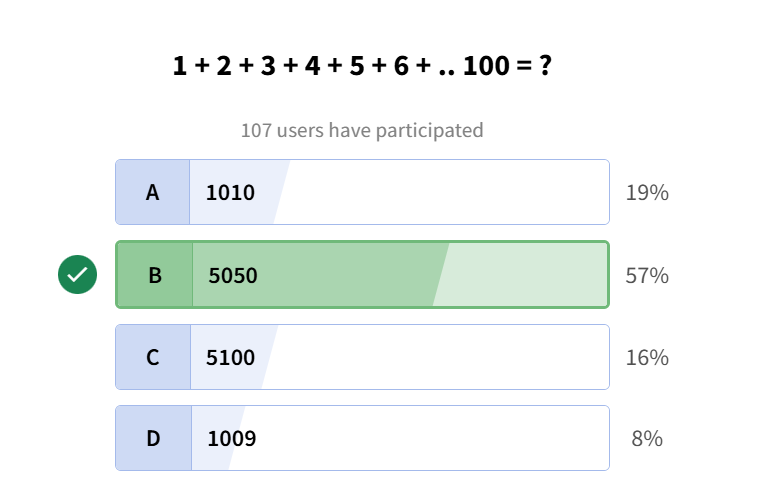


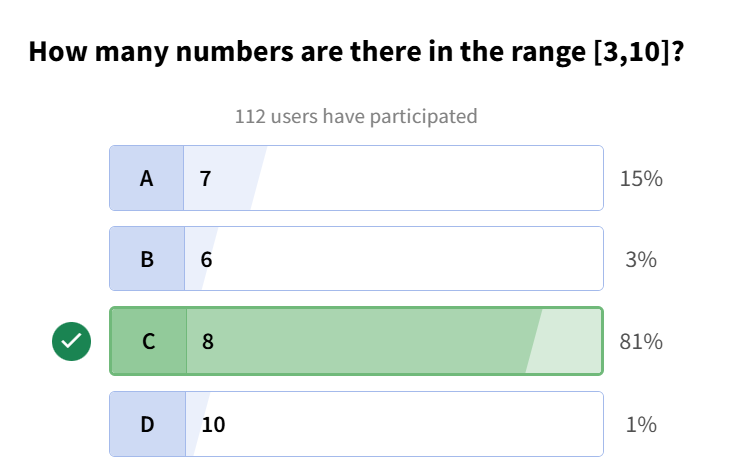
Quizzes:

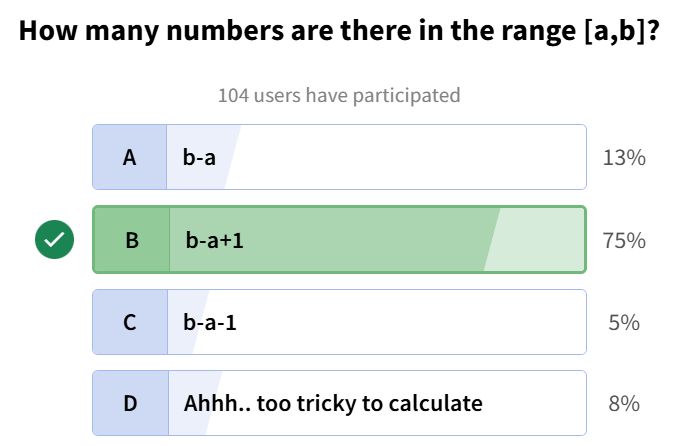


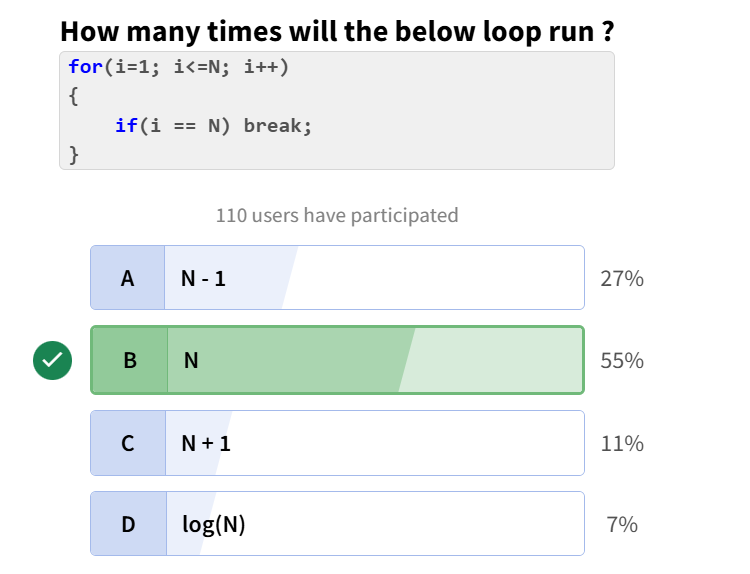


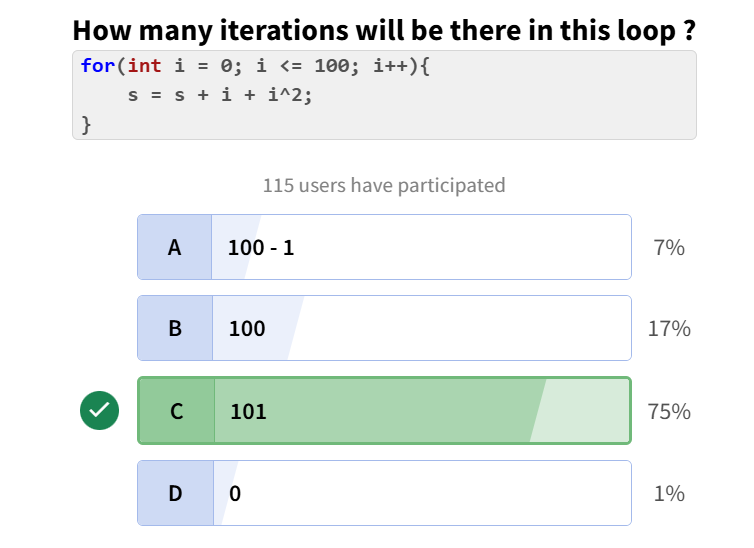


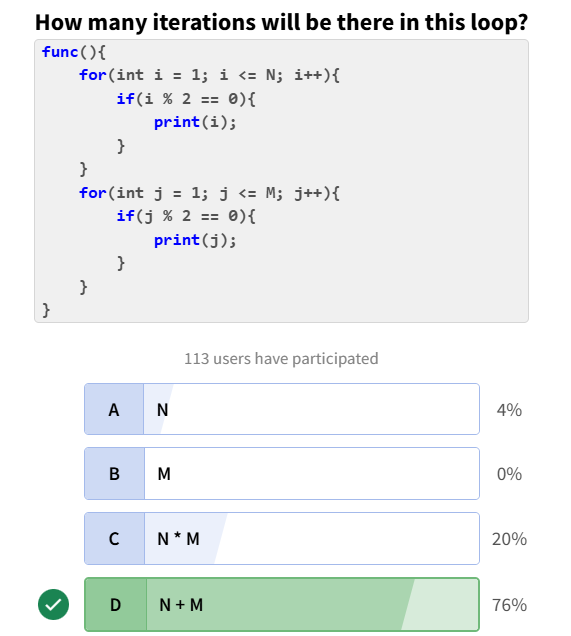












Assignments:

**Q1. Count Factors - 2**

**Problem Description**

Given an integer **A**, you need to find the count of it's factors.  
  
Factor of a number is the number which divides it perfectly leaving no remainder.  
  
Example: 1, 2, 3, 6 are factors of 6  
  
**Problem Constraints**

1 <= A <= 109  
  
**Input Format**

First and only argument is an integer A.  
**Output Format**

Return the count of factors of A.  
**Example Input**

Input 1:

5

Input 2:

10

**Example Output**

Output 1:

2

Output 2:

4

**Example Explanation**

Explanation 1:

Factors of 5 are 1 and 5.

Explanation 2:

Factors of 10 are 1, 2, 5 and 10.

CODE:

public class Solution {

    public int solve(int A) {

        int count=0;

        for(int i=1;i\*i<=A;i++){

            if(A%i==0){

                if(i\*i==A){

                    count++;

                }else{

                    count+=2;

                }

            }

        }

        return count;

    }

}

**Q2. IsPrime**

**Problem Description**

Given a number **A**. Return 1 if **A** is prime and return 0 if not.

**Note :**   
The value of **A** can cross the range of Integer.  
  
**Problem Constraints**

1 <= **A** <= 109  
  
**Input Format**

The first argument is a single integer A.  
  
**Output Format**

Return 1 if A is prime else return 0.

**Example Input**

Input 1:

A = 5

Input 2:

A = 10

**Example Output**

Output 1:

1

Output 2:

0

**Example Explanation**

Explanation 1:

5 is a prime number.

Explanation 2:

10 is not a prime number.

CODE:

public class Solution {

    public int solve(int A) {

        int count=0;

        for(int i=1;i\*i<=A;i++){

            if(A%i==0){

                if(A==i\*i)

                count++;

                else count+=2;

            }

        }

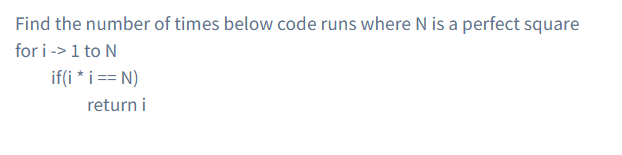
        if(count==2) return 1;

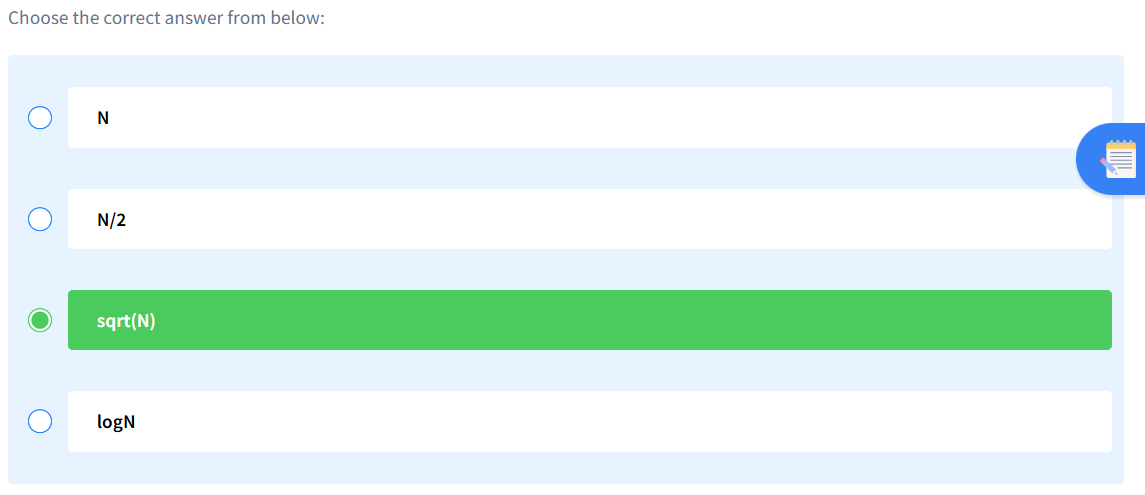
        else return 0;

    }

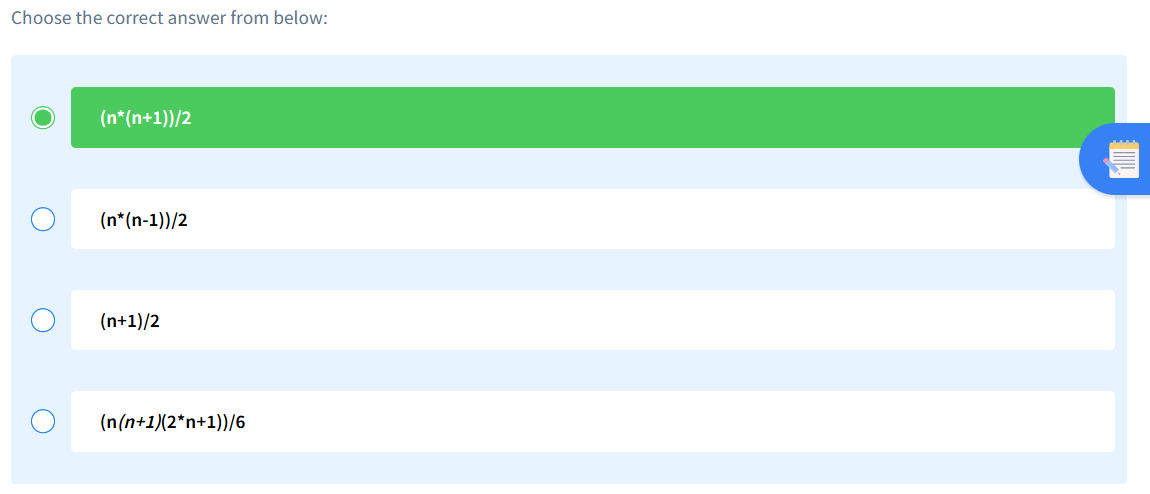
}

**Q3. Find Iterations**

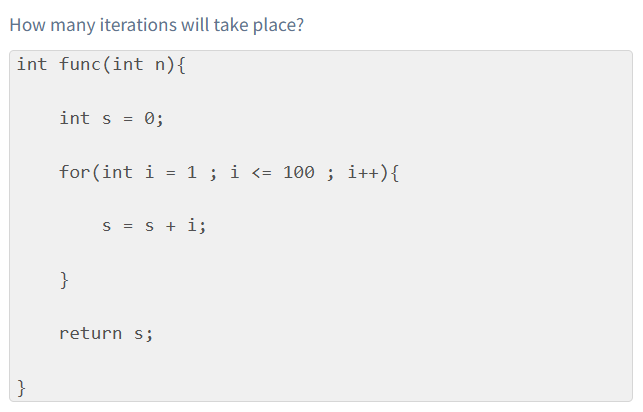


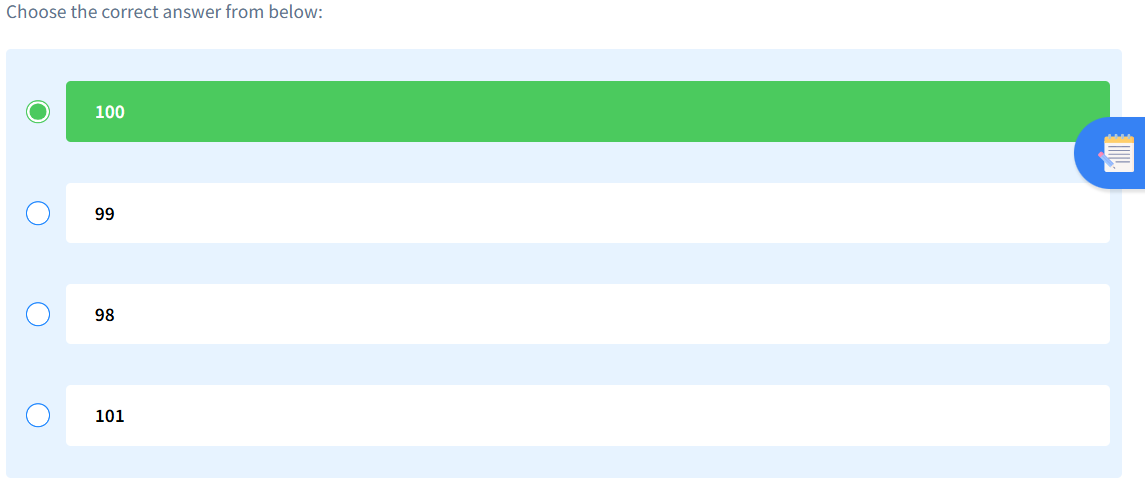


**Q4. Sum of N natural numbers**

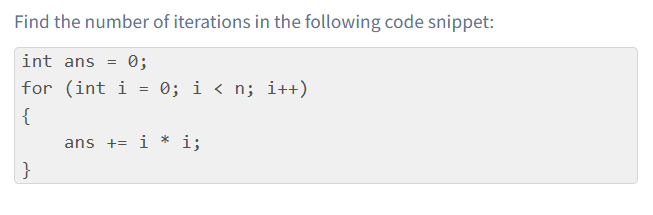


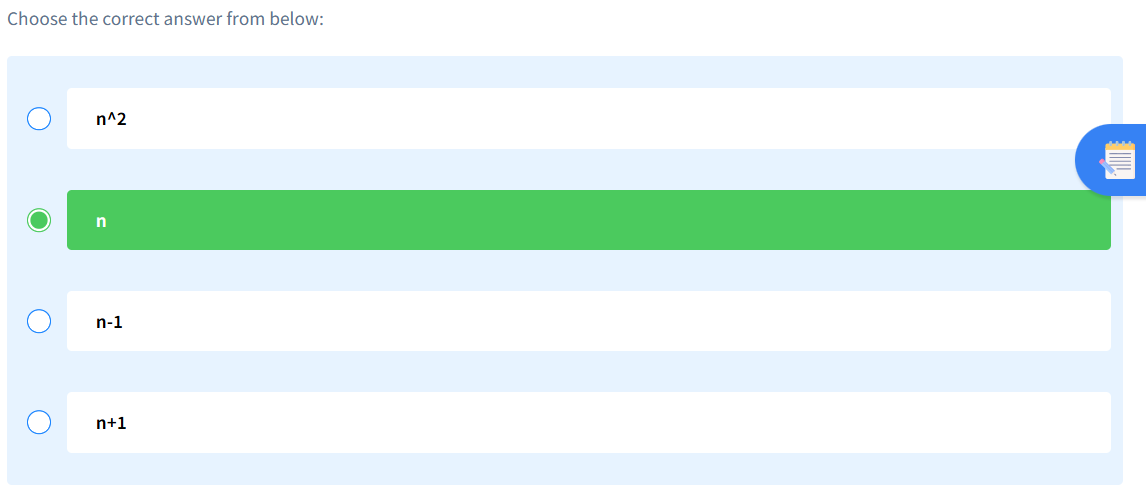
**Q5. Find Number of Iterations – 3**



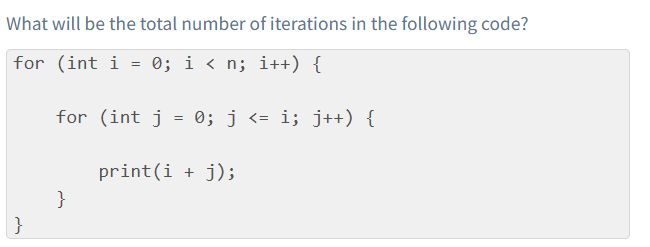


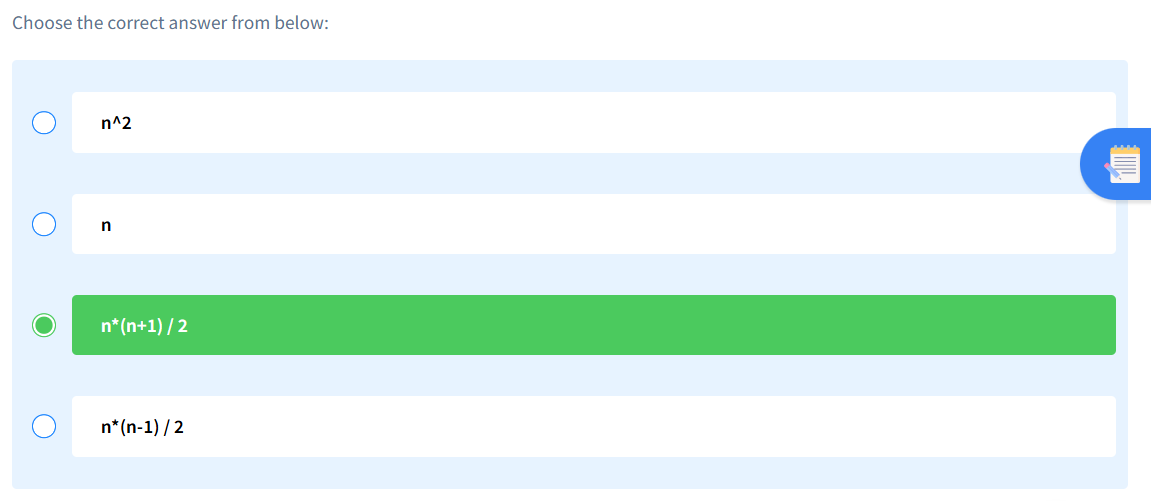
**Q6. Find Number of Iterations – i**





**Q7. Find Number of Iterations – 7**





ADDITIONAL CODING PROBLEM:

**Q1. Find Perfect Numbers**

**Problem Description**

You are given an integer **A**. You have to tell whether it is a perfect number or not.

**Perfect number** is a positive integer which is equal to the sum of its proper positive divisors.

A **proper divisor** of a natural number is the divisor that is strictly less than the number.  
**Problem Constraints**

1 <= **A** <= 106  
**Input Format**

First and only argument contains a single positive integer **A**.  
**Output Format**

Return 1 if A is a perfect number and 0 otherwise.  
  
**Example Input**

Input 1:

A = 4

Input 2:

A = 6

**Example Output**

Output 1:

0

Output 2:

1

**Example Explanation**

Explanation 1:

For A = 4, the sum of its proper divisors = 1 + 2 = 3, is not equal to 4.

Explanation 2:

For A = 6, the sum of its proper divisors = 1 + 2 + 3 = 6, is equal to 6.

CODE:

public class Solution {

    public int solve(int A) {

        int sum=0;

        for(int i=1;i\*i<=A;i++){

            if(A%i==0){

                if(i\*i==A){

                    if(i<A)

                    sum+=i;

                }else{

                    if(i<A && i!=1)

                    sum=sum+i+(A/i);

                    else

                    sum+=i;

                }

            }

        }

        if(sum==A)

        return 1;

        else return 0;

    }

}

**Q2. Count of primes**

**Problem Description**

You will be given an integer n. You need to return the count of prime numbers less than or equal to n.

**Problem Constraints**

0 <= n <= 10^3

**Input Format**

Single input parameter n in function.

**Output Format**

Return the count of prime numbers less than or equal to n.

**Example Input**

Input 1:

19

Input 2:

1

**Example Output**

Output 1:

8

Output 2:

0

**Example Explanation**

Explanation 1:

Primes <= 19 are 2, 3, 5, 7, 11, 13, 17, 19

Explanation 2:

There are no primes <= 1

CODE:

public class Solution {

    public int solve(int A) {

        int count=0;

        for(int i=1;i<=A;i++){

            int primeCount=0;

            for(int j=1;j\*j<=i;j++){

                if(i%j==0){

                    if(i==j\*j){

                        primeCount++;

                    }

                    else{

                        primeCount+=2;

                    }

                }

            }

            if(primeCount==2)

            count++;

        }

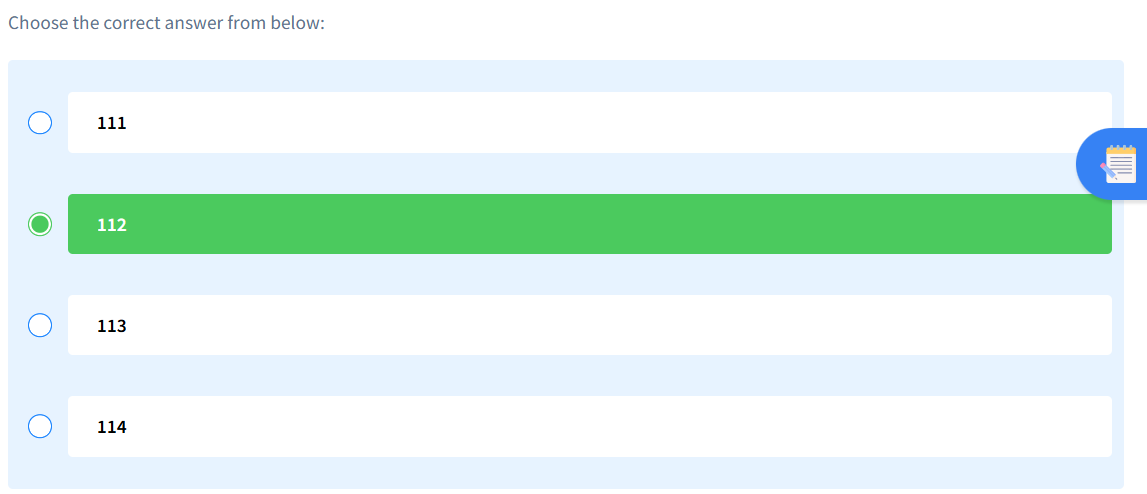
        return count;

    }

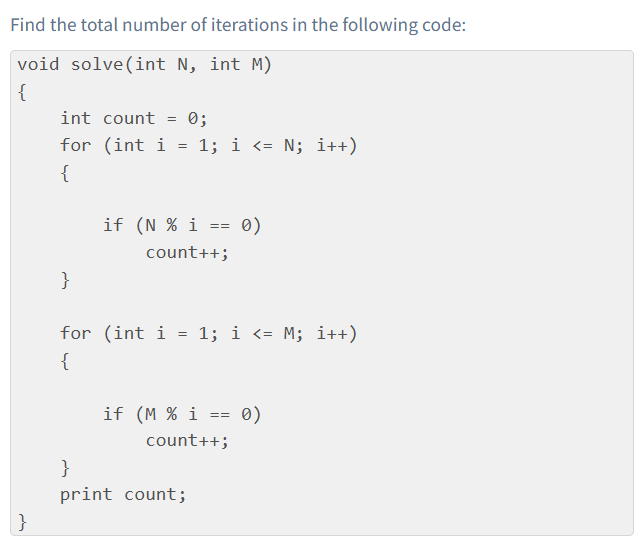
}

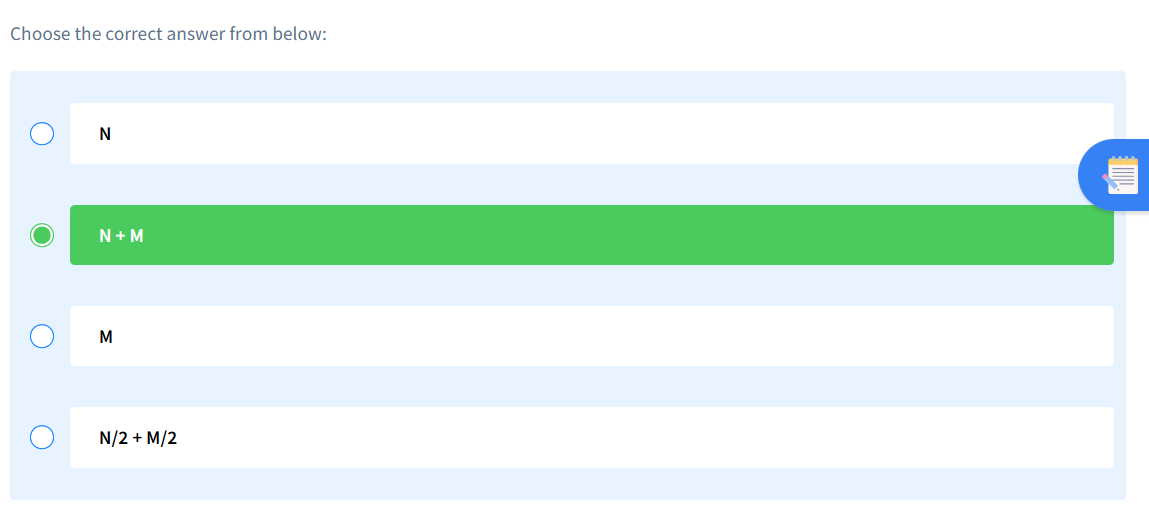
**Q3. Number of elements in a range**





**Q4. Find Number of Iterations – 2**





**Q5. Find Number of Iterations – 9**

