Qus Print all factors

12

↳ [1, 2, 3, 4, 6, 12]

n

↳ all values between 1 to n that divides my n completely.

1, 2, 3, 4, 6, 12 ... n = 13

12

i = 1 to n

12 % 1 == 0

12 % 5 = 2

12 % 12

```
public class Solution {

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

        for(int i=1;i<=n;i++){
            if(n%i==0){
                System.out.println(i);
            }
        }
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class sh
    }
}
```

1
2
3
4
6
12

Qus Print all prime factors :→

3 | 45
3 | 15
5 | 5
1

⇒ 3, 5

2 | 12
2 | 6
3 | 3
1

⇒ 2, 3

i = 2;

while(n > 0){

(n % i == 0){

syso(i);

while(n % i == 0){

n = n / i;

}

i++;

}

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

    int i=2;
    while(n>1){
        if(n%i==0){
            System.out.println(i);
            while(n%i==0){
                n /=i;
            }
        }
        i++;
    }
/* Enter your code here. Read input from STDIN. Print output t
```

n = 45 15 5 1

i = 2 3 4 5 6

45%2 = 1
45%3 = 0
5%4 = 1
5%5 = 0
1%5 ≠ 0

5/5 = 1
1%5 ≠ 0

3	45
3	15
5	5
	1

ans: 3 3 5

if your n # is divisible by 2, keep on dividing by 2.

```
steps = 0;
while(n%2==0){
    n = n/2;
    step += 2;
}
```

3% steps += 5

5% step += 5;

n → 2 Step += 2;

n → 3 step += 3;

n → 5 Step += 5;

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

    int steps =scn.nextInt();

    while(n%2==0){
        n = n/2;
        steps = steps +2;
    }

    while(n%3==0){
        n = n/3;
        steps = steps+3;
    }

    while(n%5==0){
        n = n/5;
        steps = steps+5;
    }

    System.out.println(steps);
    System.out.println(n);
    /* Enter your code here. Read input from STDIN. Print output to STDOl
}
```

n = 100 50 25 5 1

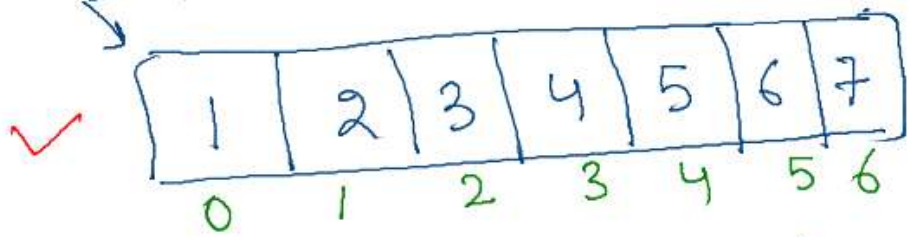
steps = 20 22 24 29 34

(n%5 ≠ 0)

false

34
1

Qus Array → array



Array is a continious memory allocation with similar data types.

1	'a'	2	3	True	5

↳ in Java

size = arr.length ⇒ 7

Syntax

int[] arr; // declaration (Array)

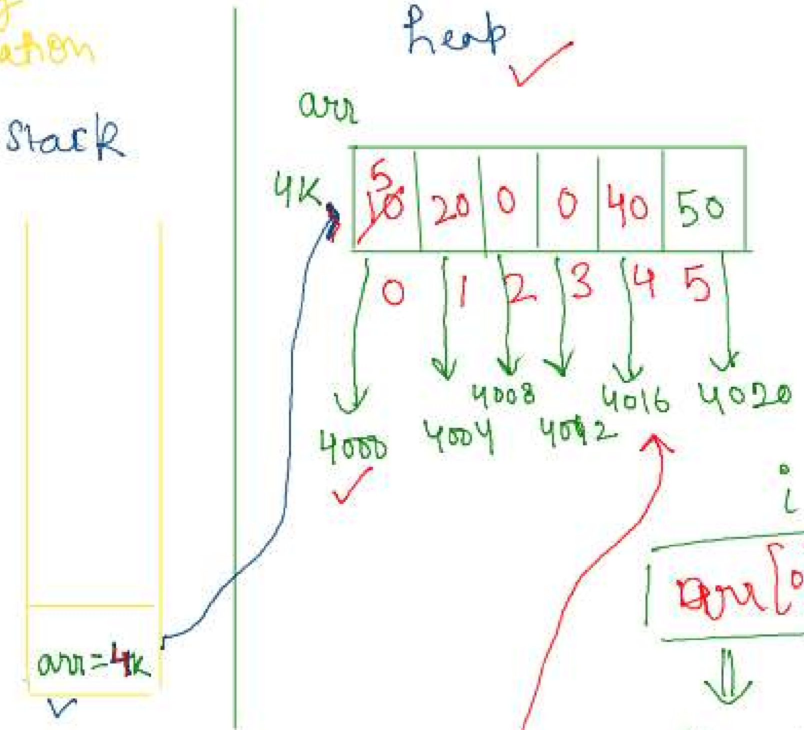
arr = new int[6]; (6 ⇒ size, n)

arr[0] = 10;

arr[1] = 20;

arr[5] = 50;

int = 4 Byte



arr[i]

i

arr[0] = 5;

⇓

4000 + 0*4

4000 = 5

arr[i] = a;

⇓

4000 + i * 4

4000 → Base address; i → index; 4 → size of datatype

arr[4] = 40 ⇒ 4000 + 4*4 = 4016

accessing elements from array is constant operation

[Base address of Array + index of (i) array * size of each element]