Note :-

$a = 2, \; b = 3$

$ans = a^b = 2^3 = 8$

```
Math.pow(a, b);   // 8
Math.max(a, b);   // 3
Math.min(a, b);   // 2
Math.abs(a);      // 2
```

_Print Armstrong in a range

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int x = scn.nextInt();
    int y = scn.nextInt();
    for (int i = x; i <= y; i++) {
        boolean ans = checkArmstrongNum(i);
        if (ans == true)
            System.out.println(i);
    }
}
public static boolean checkArmstrongNum(int n) {

    int temp = n;
    int digits = 0;

    while ( temp > 0 ) {
        digits++;
        temp /= 10;
    }

    temp = n;
    int ans = 0;
    while (n > 0) {
        int rem = n % 10;
        ans = ans + (int)Math.pow(rem, digits);
        n = n / 10;
    }

    if (ans == temp) {
        return true;
    } else {
        return false;
    }
}
```

Print all unique prime factors

$n = 240$

2	240
2	120
2	60
2	30
3	15
5	5
	1

✓ factors = 2, 2, 2, 2, 3, 5

✓ prime factor = 2, 2, 2, 2, 3, 5

✓ Unique prime factor = 2, 3, 5

n = 45) factors :- 1, 3, 5, 9, 15, 45

Prime factors :- 3, 5

Unique :- 3, 5

Code

```
for (int i=1; i<=n; i++){
    if (n%i==0){
        isPrime(i);
    }
}
```

code

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

public static void uniquePrimeFactors(int n) {
    for (int i = 2; i <= n; i++) {
        if ( n % i == 0 ) {
            boolean ans = isPrime(i);
            if (ans == true) {
                System.out.println(i);
            }
        }
    }
}
public static boolean isPrime(int n) {
    for (int i = 2; i <= n - 1; i++) {
        if ( n % i == 0 ) {
            return false;
        }
    }
    return true;
}
```

Divide n by 2 3 5 and tell steps

Take a natural number **n** as an integer input, and variable steps of integer type as input. Then perform the following operations on it.

a. If the number is **divisible** by **2**, then keep on **dividing** the number **n** by **2**, till the time the number is **divisible by 2** and also **increment** the variable steps by **2**, each time you divide the number by **2**.

b. Also, check If the number is **divisible** by **3**, then keep on dividing the number **n** by **3**, till the time the number is **divisible** by **3** and also increment the variable steps by **3**, each time you **divide** the number by **3**.

c. Also, If the number is **divisible** by **5**, then keep on **dividing** the number **n** by **5**, till the time the number is **visible** by **5** and also **increment** the variable steps by **5**, each time you divide the number by **5**.

In the end print the value of the variable steps in the first line and final value of number **n** in the second line.

n = 210
steps = 7

2	210
3	105
5	35
	7

steps = 7 + 2 + 3 + 5

n = 2472

steps = 2

final steps = ??

remaining n value = ??

dry run

2	2472
2	1236
2	618
3	309
	103

steps = 2 + 2 + 2 + 2 + 3

n = 103

steps = 11

Code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int steps = scn.nextInt();
    divideBy235(n, steps);
}
public static void divideBy235(int n, int steps) {

    while ( n % 2 == 0 ) {
        steps += 2;
        n /= 2;
    }
    while ( n % 3 == 0 ) {
        steps += 3;
        n /= 3;
    }
    while ( n % 5 == 0 ) {
        steps += 5;
        n /= 5;
    }
    System.out.println(steps);
    System.out.println(n);
}
```

n = 1125

n = 125

n = 1

n = 9000

steps = 0+2+2+2+3+3+5+5+5

2	9000
2	4500
2	2250
3	1125
3	375
5	125
5	25
5	5
	1

⇒ Arrays (M. M. Imp)

↳ Collection of similar type of data type

↳ data is stored in continuous manner.

---

syntax

```
data_type[]  arr_name = new data_type[ sizeof_array ];
```

Ex:-)

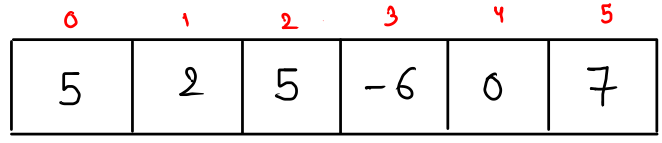
```
int[]  arr = new int[5];
```

arr =

0	0	0	0	0
0	1	2	3	4

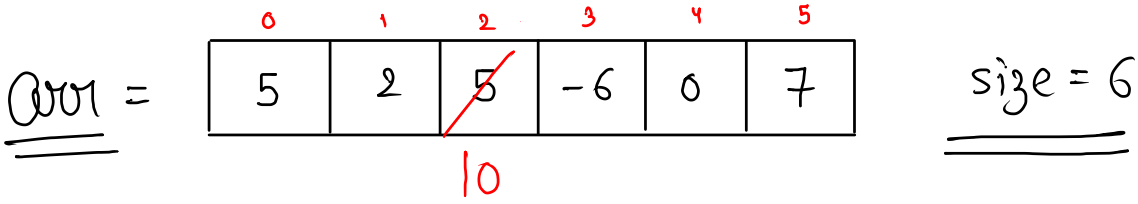
Note:- default value in int type array is always zero (only for Java)

⟹ How to access an element in array




```
int a = arr[3];
int b = arr[5];
int c = arr[6]; // Array Index out of Bound Exception
```

→ How to update an element in array



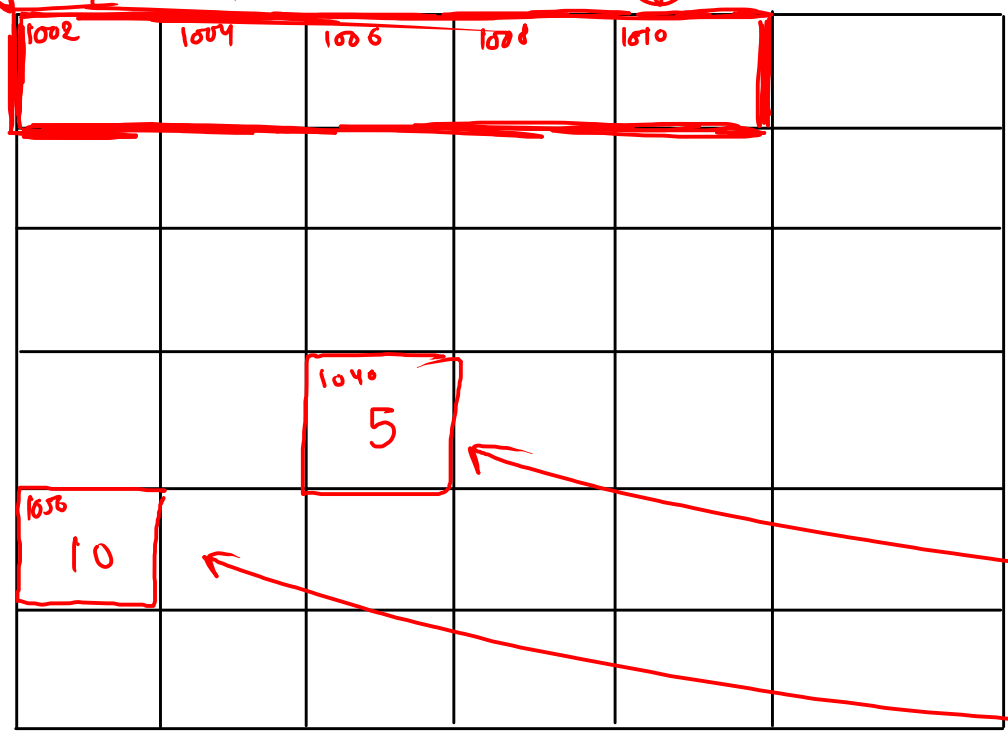

```
arr[2] = 10;
```

Note:- str.length() , arr.length

Note:- array is static in nature
(once size of array is defined then)
it cannot be changed

arr

memory allocation :-



.....∞
size

⇒ 1 TB

⇒ 1024 * 1024 * 1024 * 8 byte

int a = 5;

int b = 10;

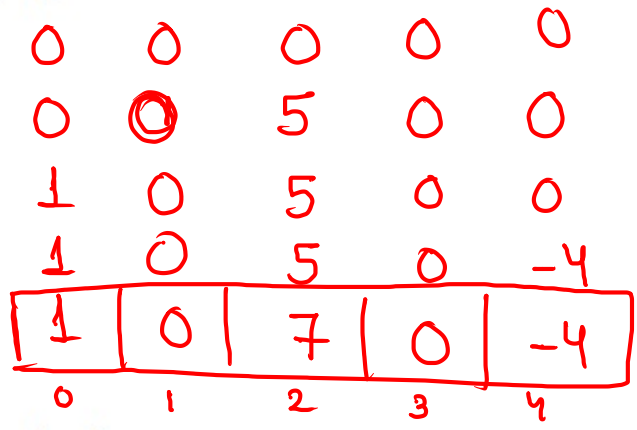
int[] arr = new int[5];

∞
size

Ex :-

```
public static void main(String[] args) {
    int[] arr = new int[5];
    arr[2] = 5;
    arr[0] = 1;
    arr[4] = -4;
    arr[2] = 7;

    for (int i = 0; i < arr.length; i++) {
        System.out.print( arr[i] + " " );
    }
}
```



Note :- { i is index
arr[i] is value at index i

Print the array elements linewise

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    for (int i = 0; i < n; i++) {
        System.out.println(arr[i]);
    }
}
```

n = 5

i = 0 , arr[0] = 100
i = 1 , arr[1] = 200
i = 2 , arr[2] = 300
i = 3 , arr[3] = 400
i = 4 , arr[4] = 500