Print the array elements linewise

Sample Input 0

```
5
1
2
3
4
5
```

Sample Output 0

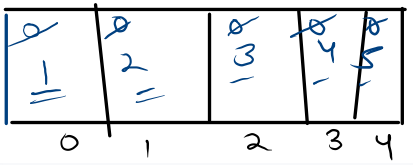
```
1
2
3
4
5
```

Take **n** as an integer input. Declare an array of size n that stores value of int data-type.

Then take **n** integer inputs and store them in the array one by one.

And print each integer in each line.

n=5



1
2
3
4
5

```java
import java.io.*;
import java.util.*;

public class Solution {

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

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

i/p

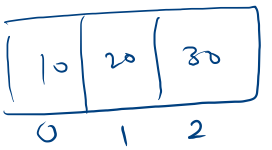
3

10

20

30

```
6      public static void main(String[] args) {
7          Scanner scn = new Scanner(System.in);
8          int n = scn.nextInt();
9          int [] A = new int[n];
10         for(int i = 0; i < n; i++){
11             A[i] = scn.nextInt();
12         }
13     }
14 }
```



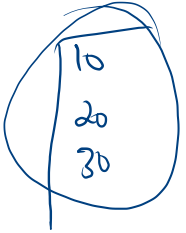

```
14         for(int i = 0; i < n; i++){
15             System.out.println(A[i]);
16         }
17     }
```

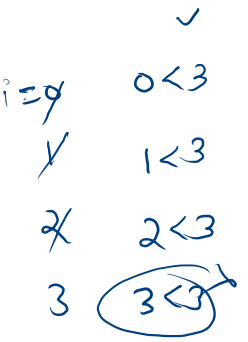
o/p

10

20

30

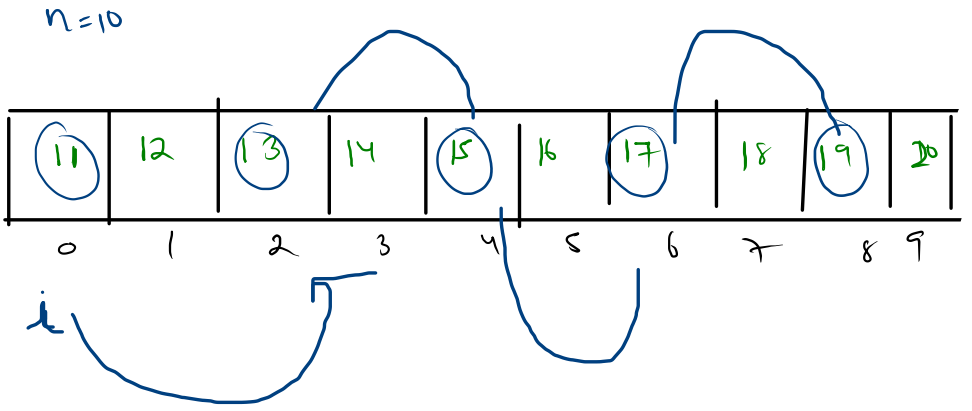




Print Alternate Array Elements Linewise

Sample Input 0

```
10
1
2
3
4
5
6
7
8
9
10
```



Sample Output 0

```
1
3
5
7
9
```

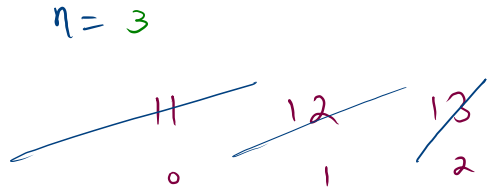
11

13

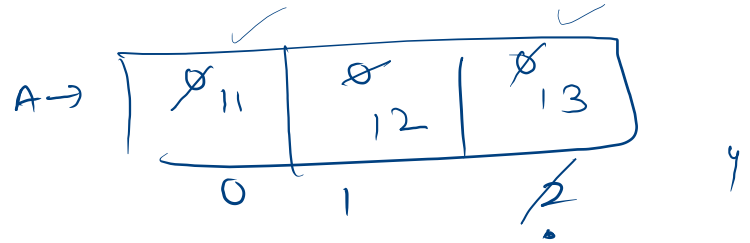
15

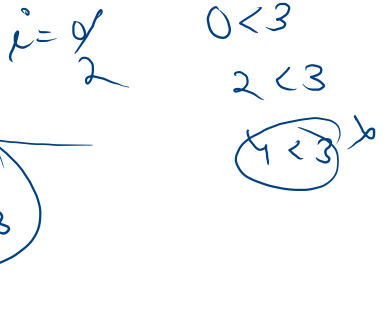
17

19

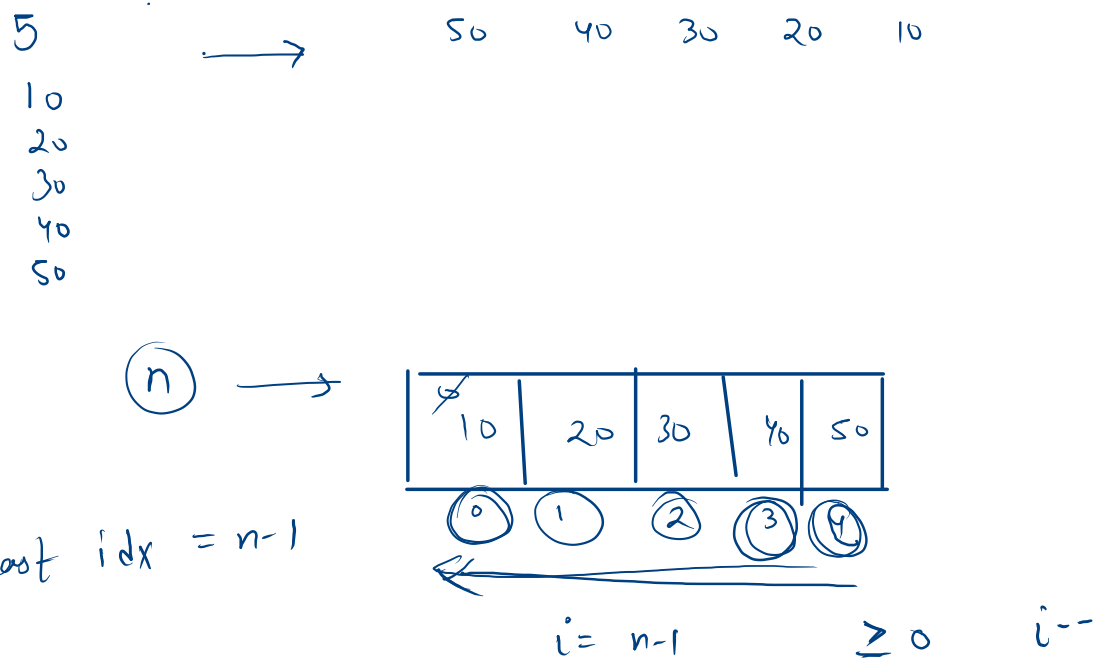



```java
import java.io.*;
import java.util.*;

public class Solution {

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

        for(int i = 0; i < n; i+=2){
            System.out.println(A[i]);
        }
    }
}
```





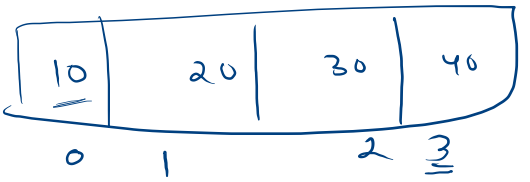
Print Array Elements Reverse linewise



A[-2]

i--

n = 4




```java
import java.io.*;
import java.util.*;

public class Solution {

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

        //output
        for(int i = n-1; i >= 0; i--){
            System.out.print(A[i] + " ");
        }
    }
}
```

i = 3

2

1

0

-1

i >= 0

3 >= 0 ✓

2 >= 0

1 >= 0 ✓

0 >= 0 ✓

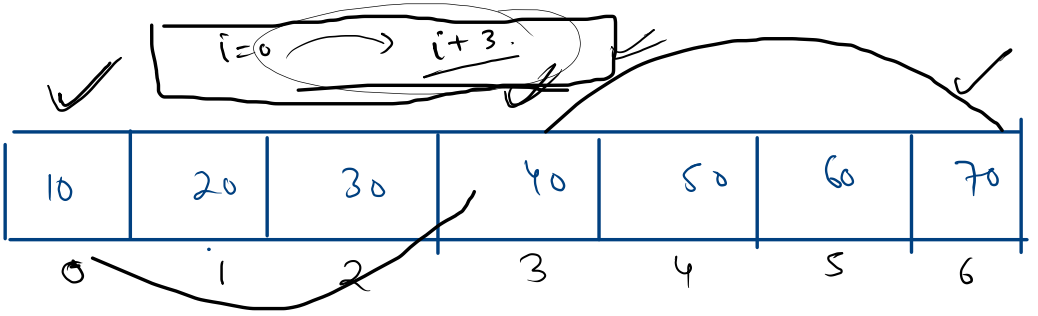
-1 >= 0 ✗

40 30 20 10

Print Array element if index divisible by 3

eg

n = 7



if (i % 3 == 0)

↳ print

```java
import java.io.*;
import java.util.*;

public class Solution {

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

        //output
        for(int i = 0; i < n; i++){
            if(i % 3 == 0){
                System.out.print(A[i] + " ");
            }
        }
    }
}
```

Check if two arrays are identical?

Take **n** as an integer input. Declare the **first** array of size **n** that stores values of **int** data-type. Then take **n** integer inputs and store them in the array one by one.

Declare the **second** array of size **m** that stores values of **int** data-type. Then take **m** integer inputs and store them in the array one by one.

Then print **true** if the arrays are equal and print **false** if the array is not equal.

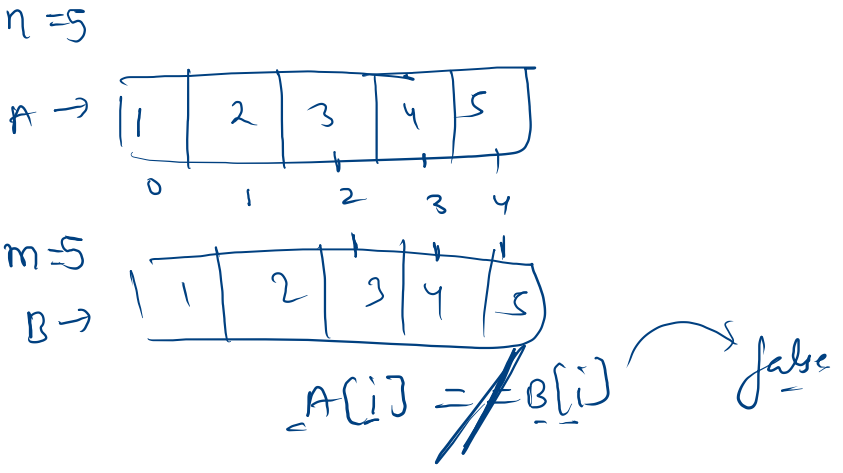
Definition of Equal Arrays: Arrays whose size is equal and whose elements at the corresponding indexes are the same

Sample Input 0

```
5
1 2 3 4 5
5
1 2 3 4 5
```

Sample Output 0

```
true
```

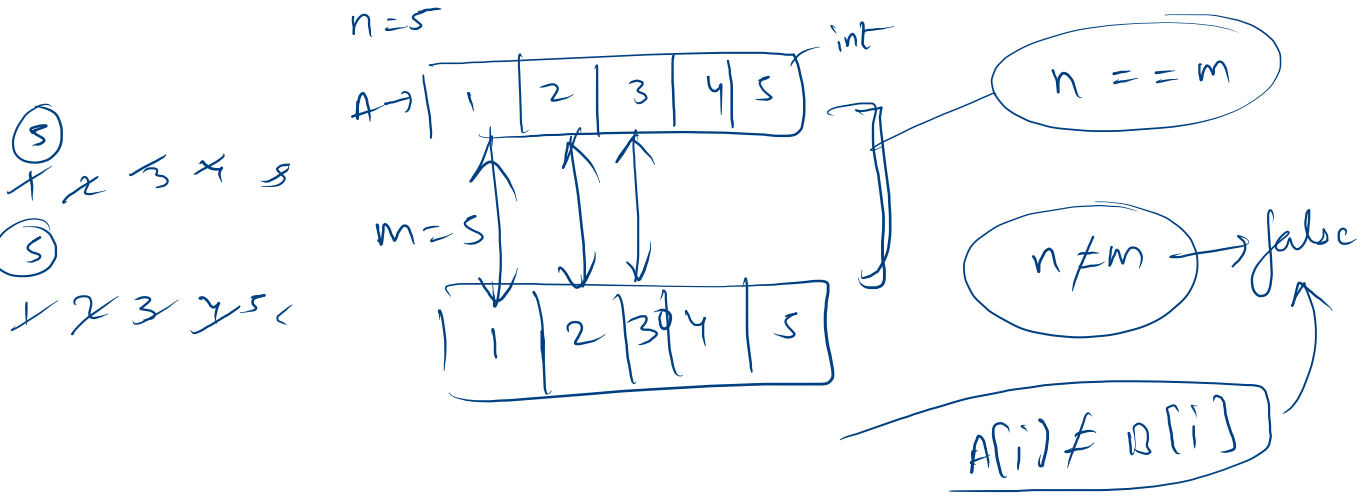


Take **n** as an integer input. Declare the **first** array of size **n** that stores values of **int** data-type. Then take **n** integer inputs and store them in the array one by one.

Declare the **second** array of size **m** that stores values of **int** data-type. Then take **m** integer inputs and store them in the array one by one.

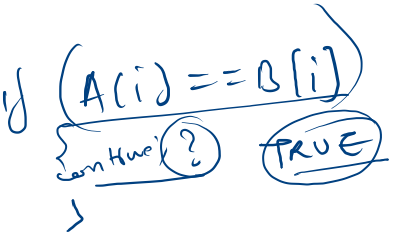
Then print **true** if the arrays are equal and print **false** if the array is not equal.

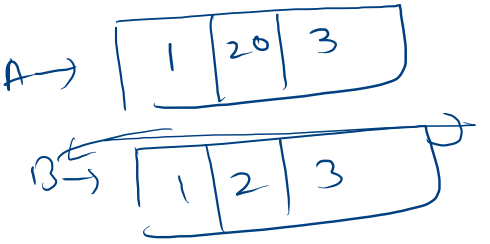
Definition of Equal Arrays: Arrays whose size is equal and whose elements at the corresponding indexes are the same



```java
import java.io.*;
import java.util.*;

public class Solution {
    public static boolean isEqual(int [] A, int [] B, int n, int m){
        if(n != m){
           return false;
        }

        //here means n and m are equal
        for(int i = 0; i < n; i++){
            if(A[i] != B[i]){
                return false;
            }
        }

        return true;
    }

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

        for(int i = 0; i < m; i++){
            B[i] = scn.nextInt();
        }
        boolean ans = isEqual(A, B, n, m);
        System.out.println(ans);

    }
```

```java
public class Solution {
    public static boolean isEqual(int [] A, int [] B, int n, int m){
        if(n != m){
           return false;
        }

        //here means n and m are equal
        for(int i = 0; i < n; i++){
            if(A[i] != B[i]){
                return false;
            }
        }

        return true;
    }
```





n → 3

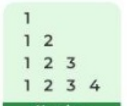
m → 3

~~n != m~~

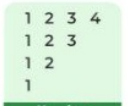
i = 0 A[0] != B[0]

Square Hollow Pattern

Number Triangular



Number Increasing Pyramid



Number Increasing Reverse Pyramid

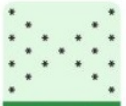
Number Changing Pyramid

Zero-One Triangle

Palindrome Triangular

Rhombus Pattern

Diamond Pattern



Butterfly Star Pattern

Square Fill Pattern

Right Half Pyramid

Reverse Right Half Pyramid

Left Half Pyramid

Reverse Left Half Pyramid

K Pattern

Triangle Star Pattern

Reverse Number Triangle Pattern

Mirror Image Triangle Pattern

Hollow Triangle Pattern

Hollow Reverse Triangle Pattern

Hollow Diamond Pyramid

Hollow Hourglass Pattern

Pascal's Triangle

Right Pascal's Triangle