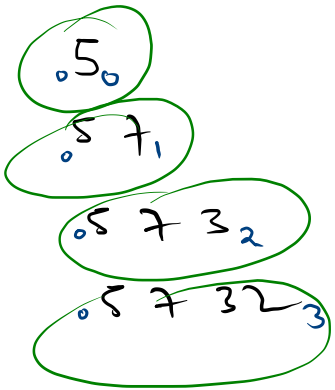
Revision.

5	7	3	2
0	1	2	3



7 1
7 3 2
7 32 3
3 2 2
3 2 3
2 3

```
for ( start = 0 ---  start < n ; start++)
{  for (end = start + ..  end < n , end++)
   {
   }
```

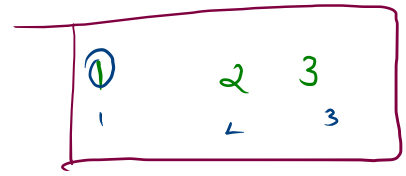
— start / end.

Press Esc to exit full screen

```java
import java.util.*;
public class Main {
    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();
        }
        boolean ans = false;

        for(int start = 0; start < n; start++){
            for(int end = start; end < n; end++){
                //one sub array
                int sum = 0;
                for(int k = start; k <= end; k++){
                    sum += A[k];
                }
                if(sum == 0){
                    ans = true;
                }
            }
        }

        System.out.println(ans);
    }
}
```



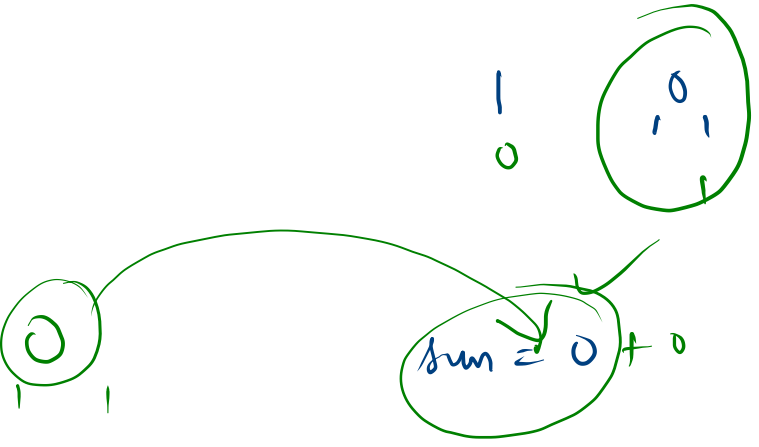
ans = false. true

start = 0
end = 0 1

sum = 0 -1 0 =

k = 0 1

start = 1 end = 3



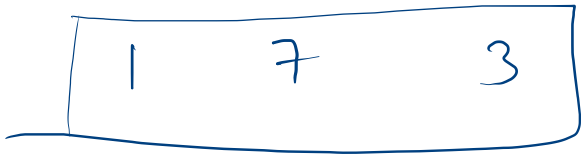
for (s —

if (!m==0)

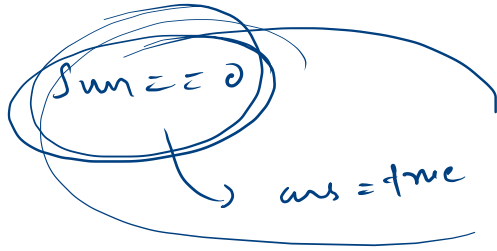
{

+me.

}

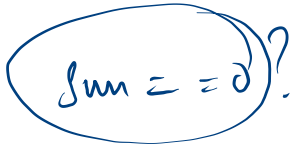


?



max

false



Max Subarray

Given an array Arr[] of N integers. Find the contiguous sub-array(containing at least one number) which has the maximum sum

Problem Statement

Samantha is a college student who is struggling to balance her part-time job with her studies. One day, she decided to take a break and went to the nearby park. While sitting on the bench, she overheard a group of students discussing a coding challenge they were trying to solve. Samantha was intrigued and asked them about the challenge.

The challenge was to find the **contiguous sub-array** with the **maximum sum** from a given array. Samantha decided to take up the challenge and spent the next few hours working on it. Finally, she was able to come up with a solution that could find the **maximum sum sub-array in linear time.**

count = $\frac{n(n+1)}{2}$ = 15

-1

-1

-1 2

4

-1 2 3

-1 2 3 -2

-1 2 3 -2 1

3

3

3 - 2

3 -2 1

2

2 3

5

2 3 -2

2 3 -2 1

-2 1

Input:

```
5
-1 2 3 -2 1
```

Output:

5

Kadane's Algo

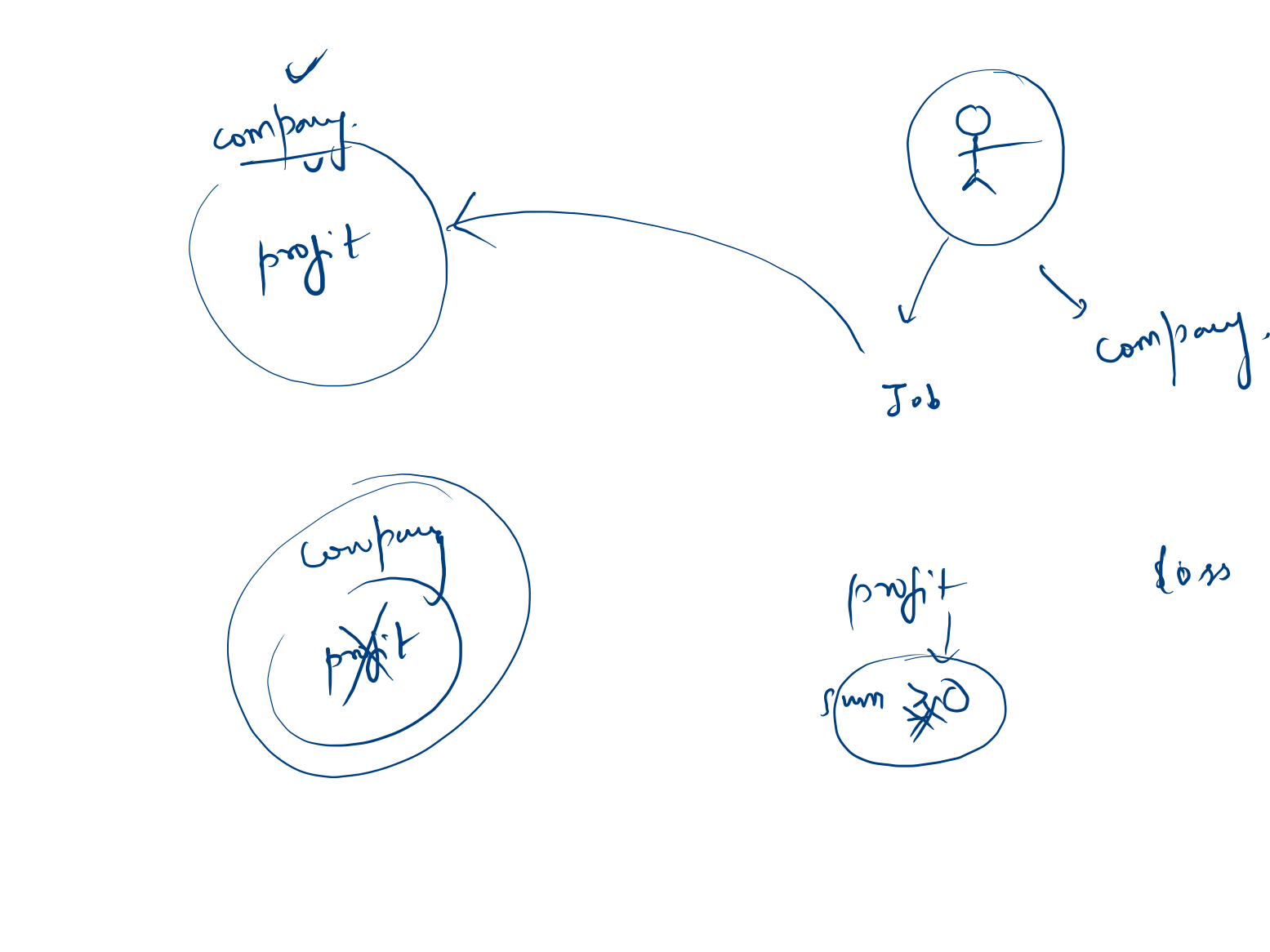
max sum of all subarrays

ans = 0

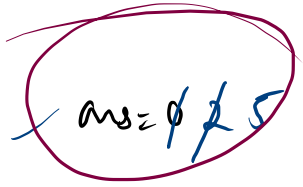
sum = 0 → curr status

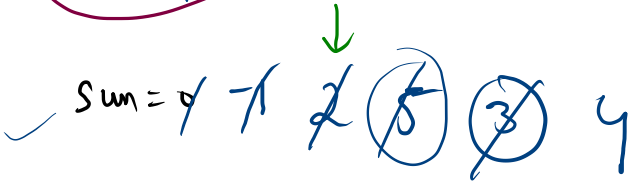
2	-1	-3	1	2
0	1	2	3	4

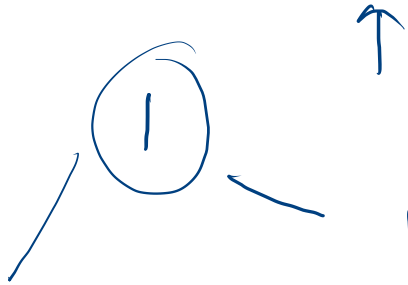
sum = ~~0~~ ~~2~~ ~~1~~ ~~-2~~ ~~1~~ (3) ✓


company
profit
Job
company
company
profit
profit
sum
loss

Kadane's Algo.







Input:

```
5
-1 2 3 -2 1
```

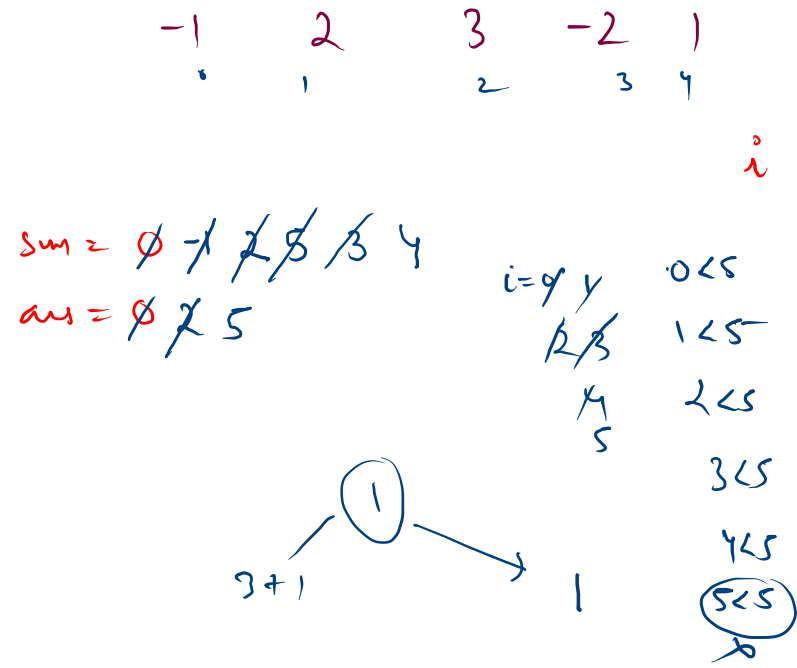
Output:

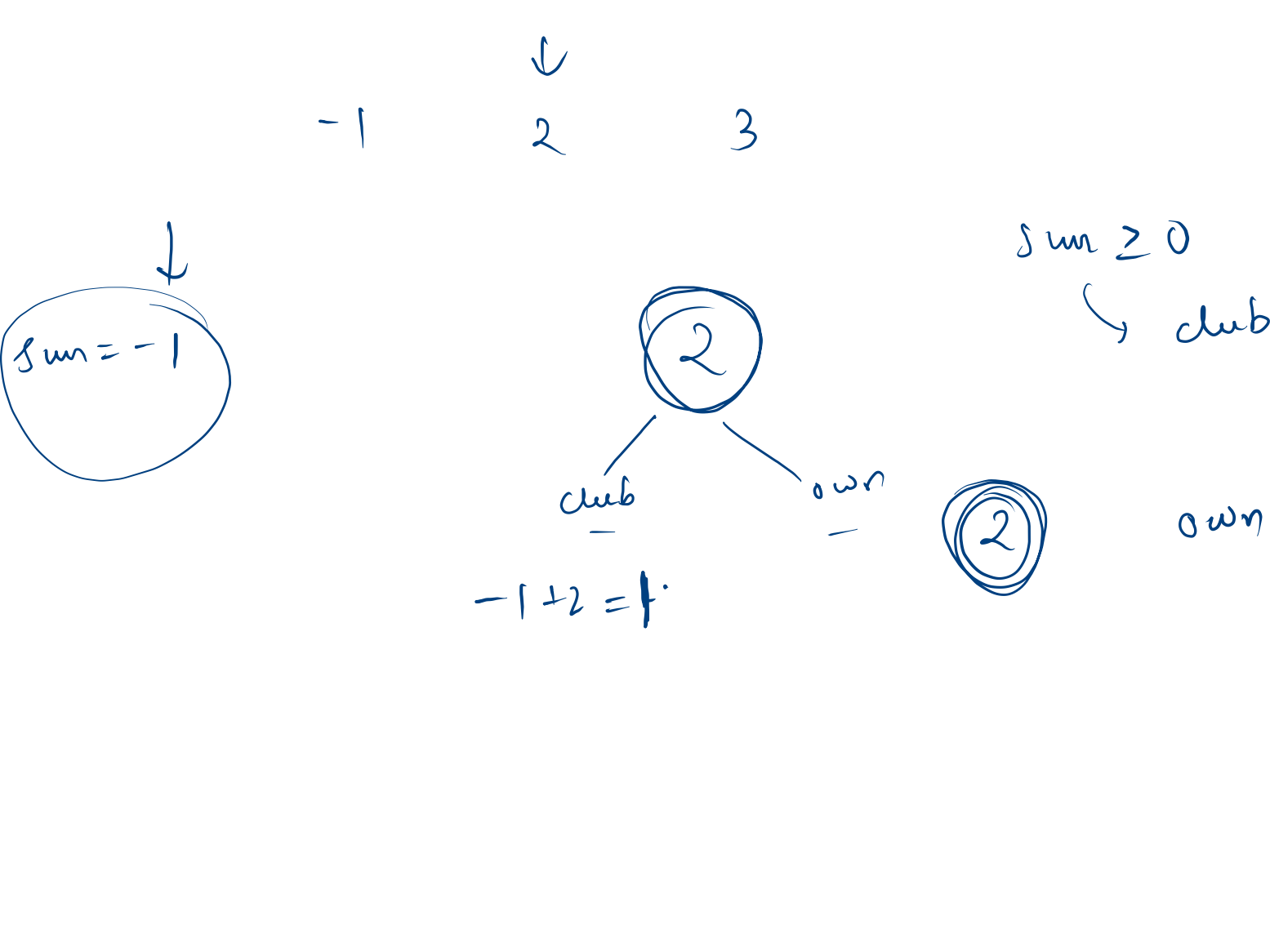
```
5
```

```java
import java.util.*;

public class Main {

    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 ans = 0;
        int sum = 0;
        for(int i = 0; i < n; i++){
           if(sum > 0){
            sum += A[i];
           }
           else{
            sum = A[i];
           }
           ans = Math.max(ans, sum);
        }
        System.out.println(ans);
    }
}
```




-1
2
3
sum = -1
2
club
own
-1+2 = 1
sum ≥ 0
club
2
own

$$\begin{pmatrix} -2 & 2 \end{pmatrix} \quad 0 \quad 4$$

Reverse Array

Reverse array

Problem Statement

Sophia was an aspiring programmer who had just started learning about arrays. One day, her mentor gave her a task to create an array of size **N** and reverse it. Sophia was excited to take up the challenge and started working on it immediately.

She defined an array arr[] of size **N** and filled it with integers. However, she got stuck when it came to reversing the array. Can you help Sophia with this task? Write a program to define an array of size **N**, fill it with integers, and reverse the array.

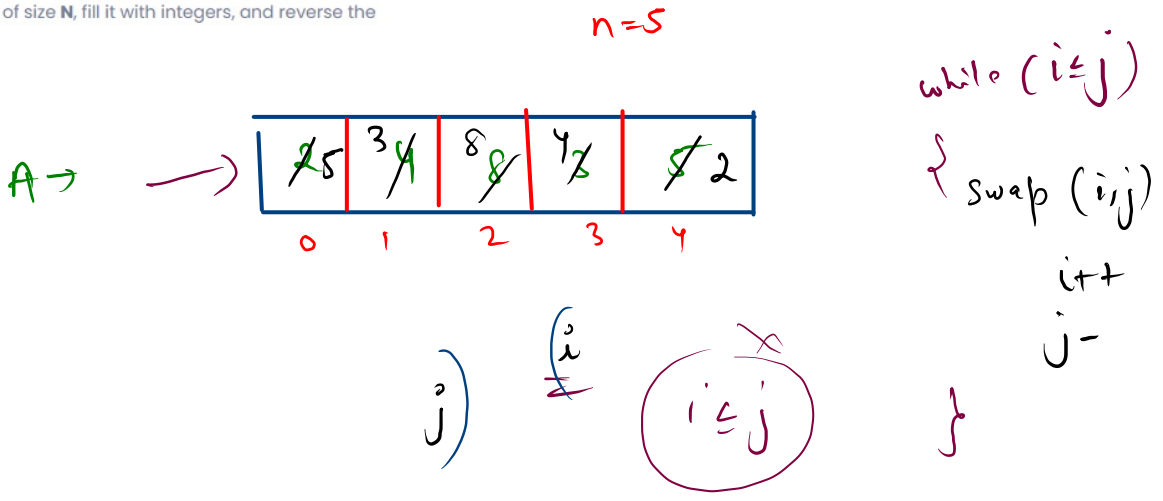
Test Case 1

Input:

```
5
2
4
8
3
5
```

Output:

```
5
3
8
4
2
```

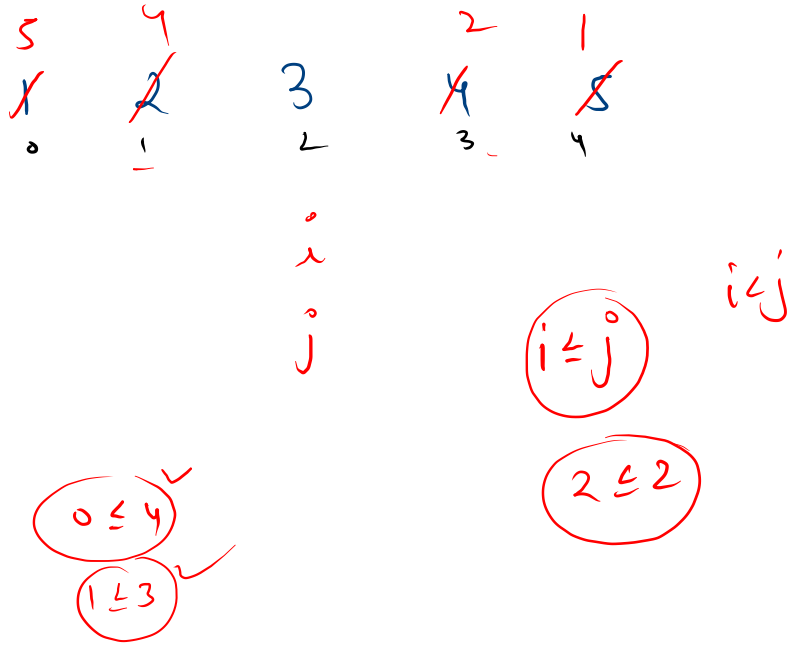


```java
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int[] A = new int[n];
        for(int i = 0; i < n; i++){
            A[i] = scn.nextInt();
        }
        int i = 0;
        int j = n-1;

        while(i <= j){
            int tmp = A[i];
            A[i] = A[j];
            A[j] = tmp;
            i++;
            j--;
        }

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

    }
}
```



Interleaving x and y Elements

Interleaving x and y Elements

Problem Statement

Suppose you have an array called nums that contains **2N** elements. The first N elements are labeled as x1, x2, ..., xn, and the remaining **N** elements are labeled as y1, y2, ..., yn. Your task is to rearrange the elements of the nums array in a specific way. Specifically, you need to create a new array where the first element is **x1**, the second element is **y1**, the third element is **x2**, the fourth element is **y2**, and so on, up to the nth element being **yn**.

In other words, you need to return an array in the form [x1, y1, x2, y2, ..., xn, yn].

n=3

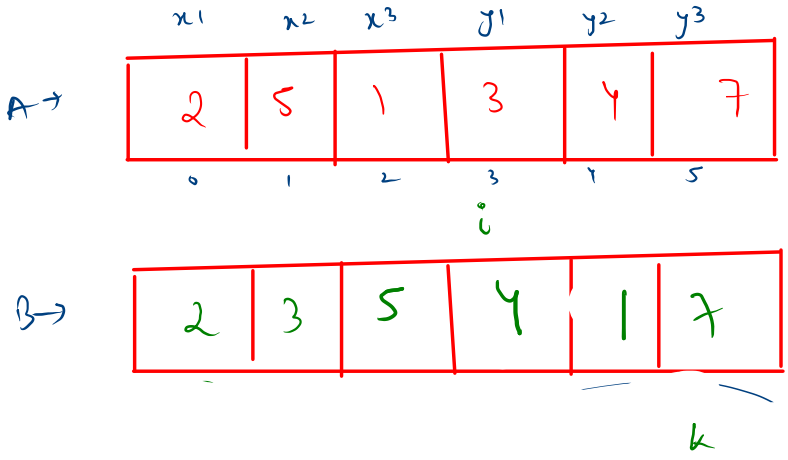
Test Case 1

Input:

6
2 5 1 3 4 7

Output:

2 3 5 4 1 7



len=6

i=0 i<len/2

i=1 i<3

i=2

1+3

2+3

B[k] = A[i]

k++

B[k] = A[i + n/2]

k++

0+3

B[k]=A[3]

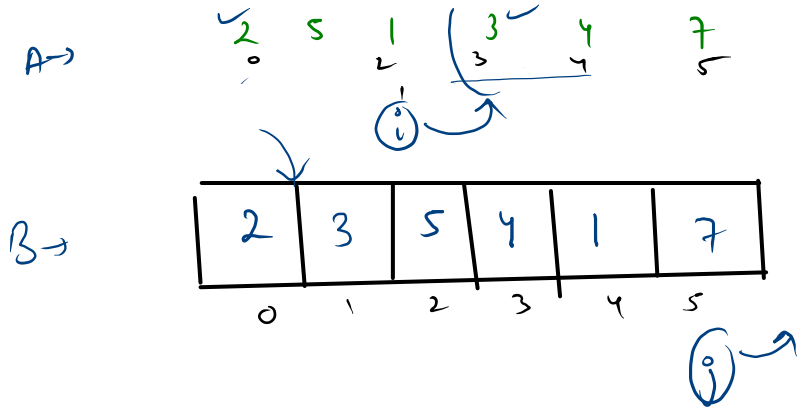
```java
import java.util.*;

public class Main {
    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 [] B = new int[n];
        int i = 0;
        int j = 0;

        while(i < A.length/2){
            B[j] = A[i];
            j++;
            B[j] = A[i+(A.length/2)];
            j++;
            i++;
        }

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



ans → 2 3 5 4 1 7