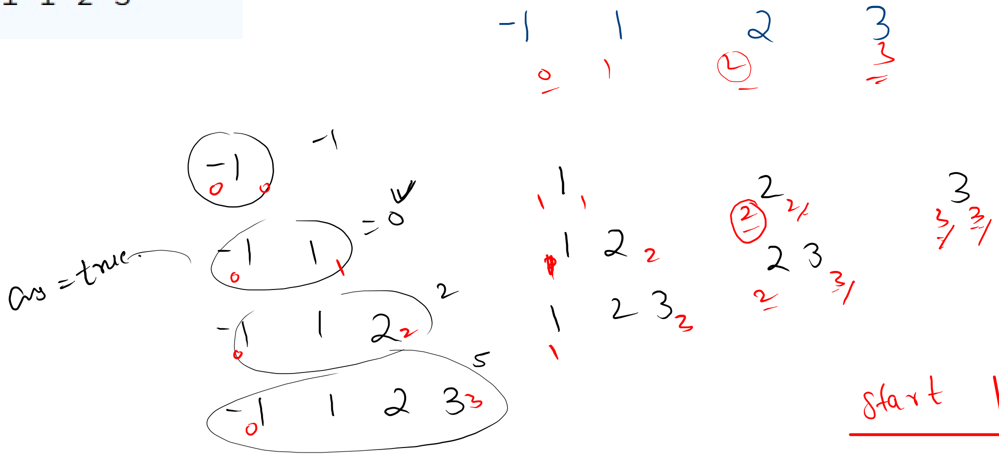
Sum Equals Zero

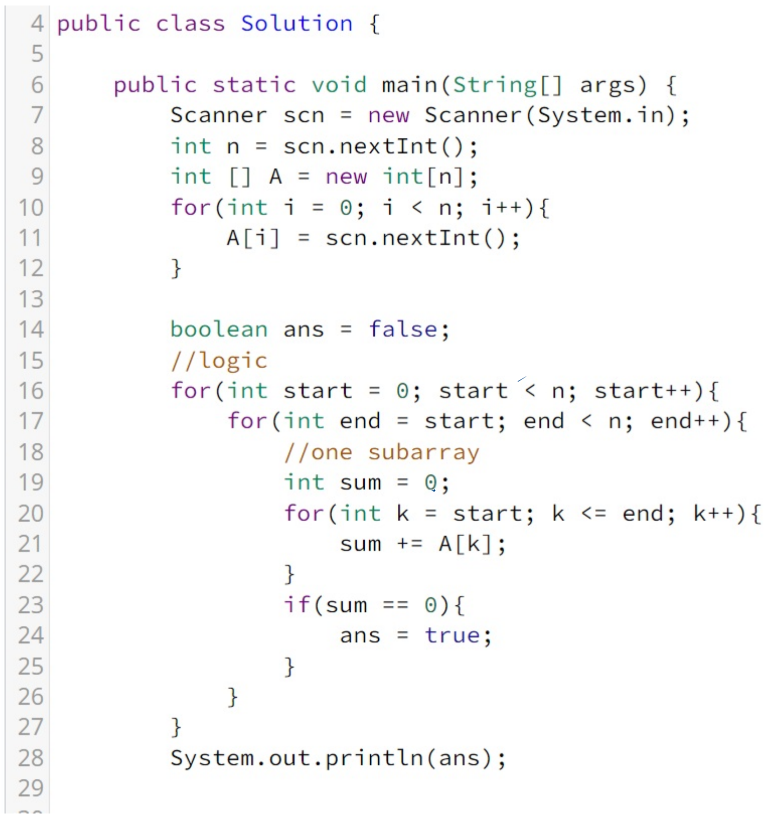
Liam is a stock trader who is analyzing the **stock prices** of a company. He has stored the stock prices in an array of size N. Liam wants to find out if there is a **subarray** of the stock prices whose sum is **zero**. If such a subarray exists, Liam can take advantage of it to make a profit.

Can you write a program to help Liam determine whether the array contains a **subarray** whose sum is **zero**?

```
4 → n
-1 1 2 3
```



start	end
0	0, 1, 2, 3
1	1, 2, 3
2	2, 3
3	3



```java
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();
        }

        boolean ans = false;
        //logic
        for(int start = 0; start < n; start++){
            for(int end = start; end < n; end++){
                //one subarray
                int sum = 0;
                for(int k = start; k <= end; k++){
                    sum += A[k];
                }
                if(sum == 0){
                    ans = true;
                }
            }
        }
        System.out.println(ans);

```

n=3

1	-1	0
0	1	2

ans = ~~false~~ true

st = 0   0<3

ed = ~~0~~ ~~1~~ 2   0<3 ✓ 1<3 ✓ 2<3 ✓

sum = ~~0~~ 1-1 = 0

k = ~~0~~ ~~1~~ 2   k≤1 0≤1 ✓ 1≤1 ✓ 2≤1 ✗

Max Subarray 2

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.

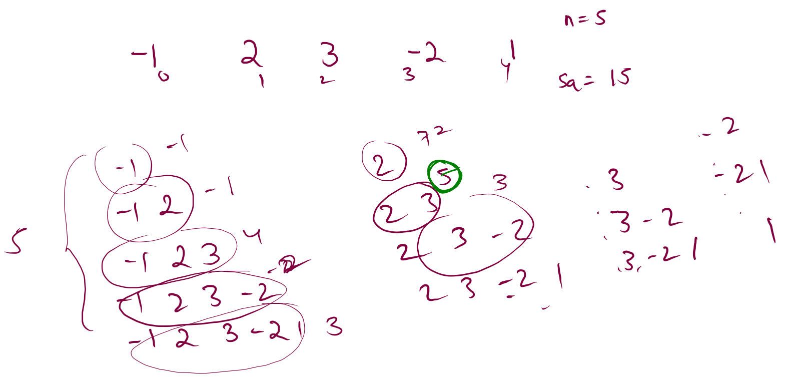
The challenge was to find the **contiguous sub-array** with the **~~maximum sum~~** from a given array. Samantha decided to take up the challenge and spent the next few hours working on it. Finally, she was able to come up with a solution that could find the **maximum sum sub-array in linear time.**

Sample Input 0

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

Sample Output 0

```
5
```



1st basic approach : brute force. approach.

$O(n^3)$

1. all subarray
↓
sum of all subarray

2.
↓
max of step 2.

3.

Kadane's Algo.

business man.

profit

start own business

acquire other business.

max = ~~0~~ ~~2~~ 5 104

-1	2	3	-2	1	100
0	1	2	3	4	5
					i

profit = ~~0~~ ~~-1~~ ~~2~~ ~~5~~ ~~3~~ 4

100

104

B

max sum of subarray.

			x		
-1	2	3	-2	1	100
0	1	2	3	4	5
			i		

max = 5

curr = ~~5~~ 3

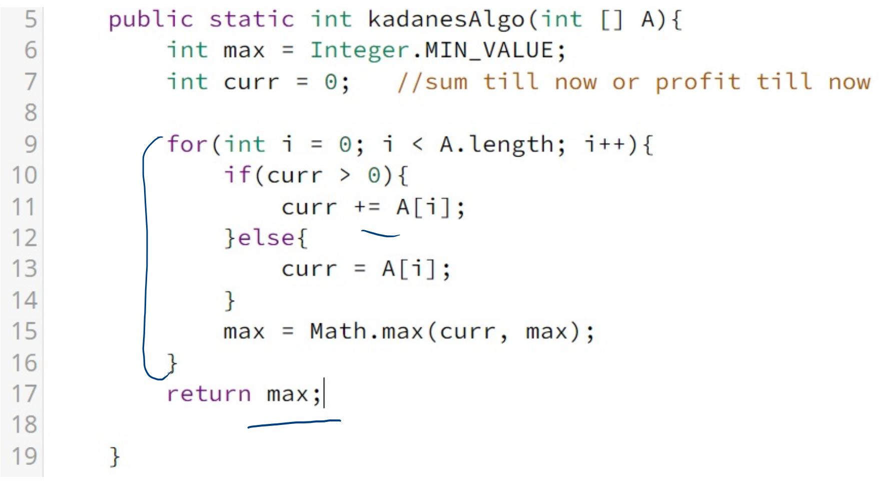
-2, $\textcircled{3}$

$max = -\cancel{\infty}\ \cancel{-1}\ \cancel{2}\ \boxed{5}$

$curr = \cancel{0}\ \cancel{-1}\ \cancel{2}\ \cancel{5}\ \cancel{3}\ 4$

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

i

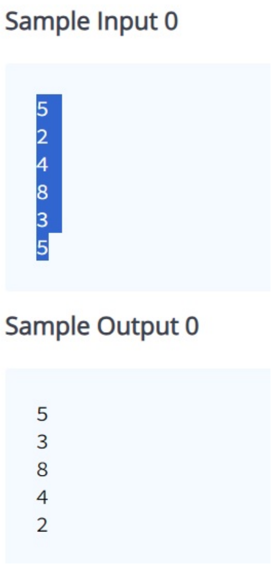


```java
public static int kadanesAlgo(int [] A){
    int max = Integer.MIN_VALUE;
    int curr = 0;   //sum till now or profit till now

    for(int i = 0; i < A.length; i++){
        if(curr > 0){
            curr += A[i];
        }else{
            curr = A[i];
        }
        max = Math.max(curr, max);
    }
    return max;

}
```

Reverse Array.



Sample Input 0

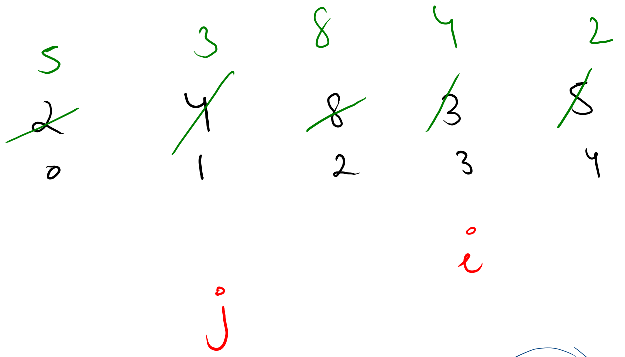
```
5
2
4
8
3
5
```

Sample Output 0

```
5
3
8
4
2
```

logic.

$n=5$



$\boxed{i<j}$ swap $i{+}{+}$ $j{-}{-}$

or

$\boxed{i \le j}$ swap $i{+}{+}$ $j{-}{-}$

Interleaving x and y Elements

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].`

NOTE :- After answering the question, attempt the related question in the linked resource to improve your understanding of this question. Click here

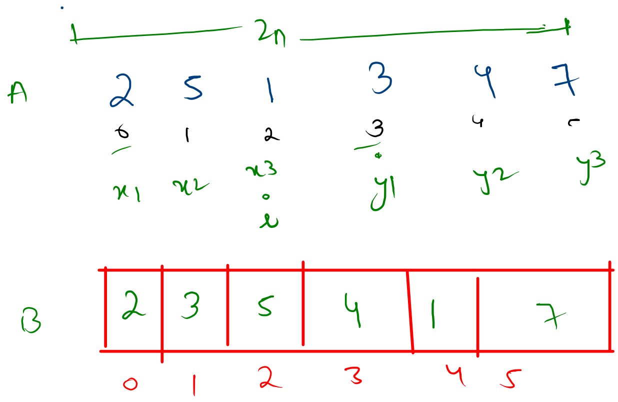
Sample Input 0

```
6
2 5 1 3 4 7
```

$n=3$

Sample Output 0

```
2 3 5 4 1 7
```



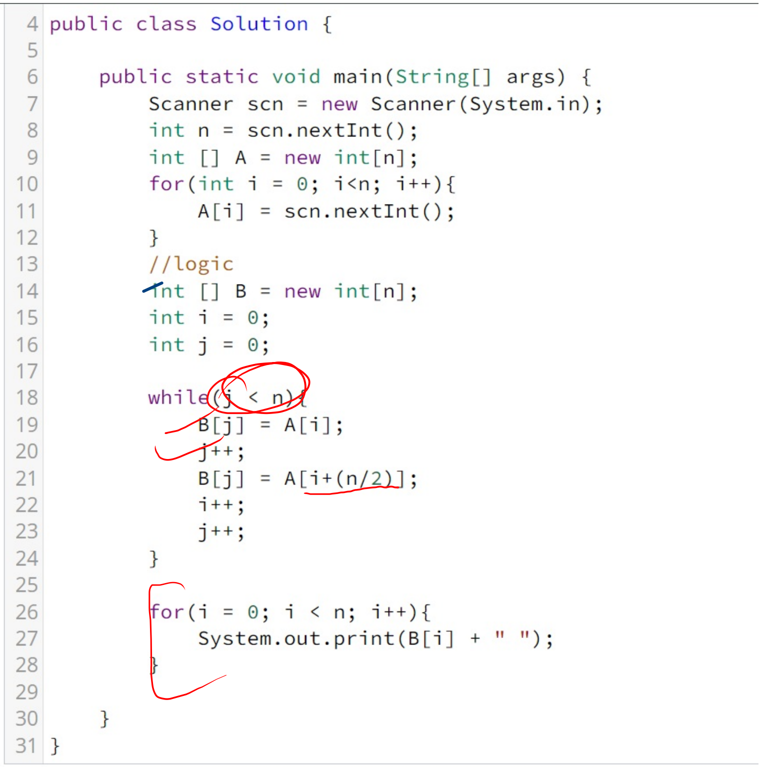
j

x_1

A[0] A[3] [A1] A[4]

B[j] = A[i] , j++

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



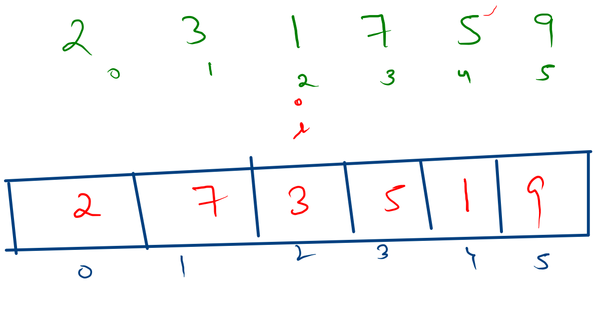
```java
public class Solution {

    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();
        }
        //logic
        int [] B = new int[n];
        int i = 0;
        int j = 0;

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

        for(i = 0; i < n; i++){
            System.out.print(B[i] + " ");
        }

    }
}
```



A[0] A[3] A[1] A[4] A[2] A[5]

$x = n/2 = \boxed{3}$

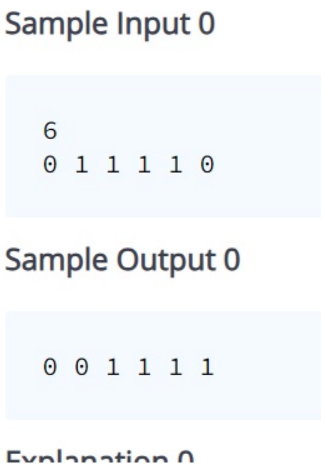
i=0

i	i+x	i	i+x	i	i+x
0		1	4	2	5

sort 0 1

0... 1...

even...odd...


Sample Input 0

```
6
0 1 1 1 1 0
```

Sample Output 0

```
0 0 1 1 1 1
```

Explanation 0

eg. $\overset{0}{\cancel{1}}\ \overset{0}{\cancel{1}}\ 0\ 0\ \overset{0}{\cancel{1}}\ \overset{1}{\cancel{0}}\ \overset{1}{\cancel{0}}\ 1\ 1\ \overset{1}{\cancel{0}}$

j

i

✓ TC = O(n) ✓

Arrays.sort

✗ TC = O(nlogn)

$\boxed{i<j}$

```
if A[i] == 0
    i++
else if A[j] == 1
    j--
else    swap(i,j)
    i++
    j--
```