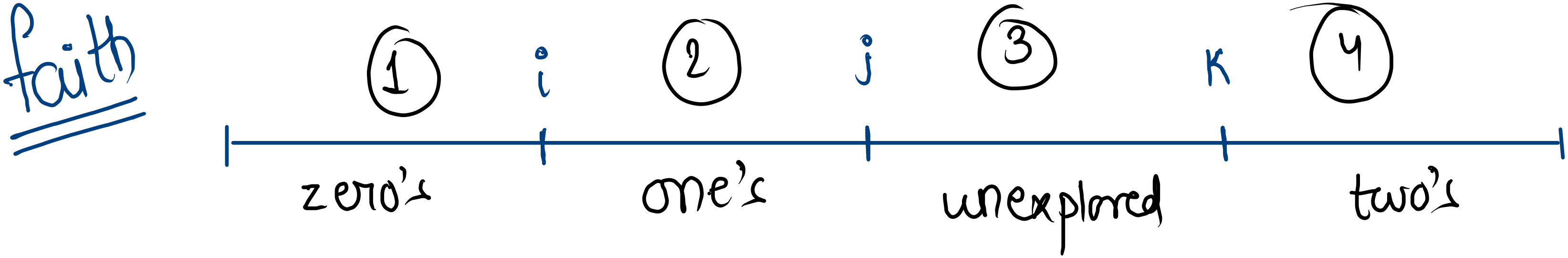
Rotate Right

1) to handle -ve k values :- $k = k+n$

2) to handle k value greater than n :- $k = k \% n$

Sort 0 1 2

	0	1	2	3	4	5	6	7	8	9	10	11
arr =	0	1	0	1	1	2	0	2	1	2	0	1

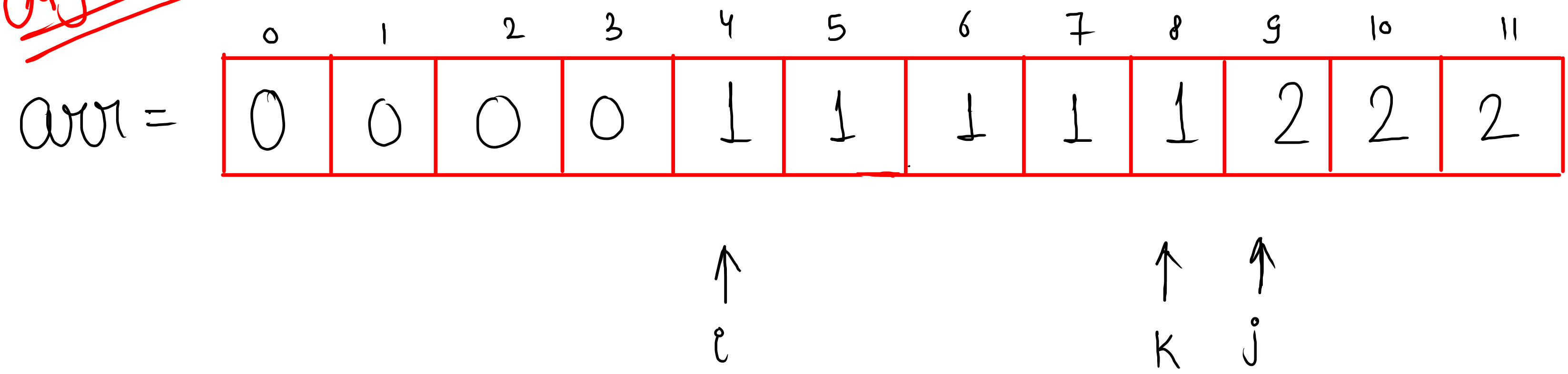


$i = 0$

$j = 0$

$K = n-1$

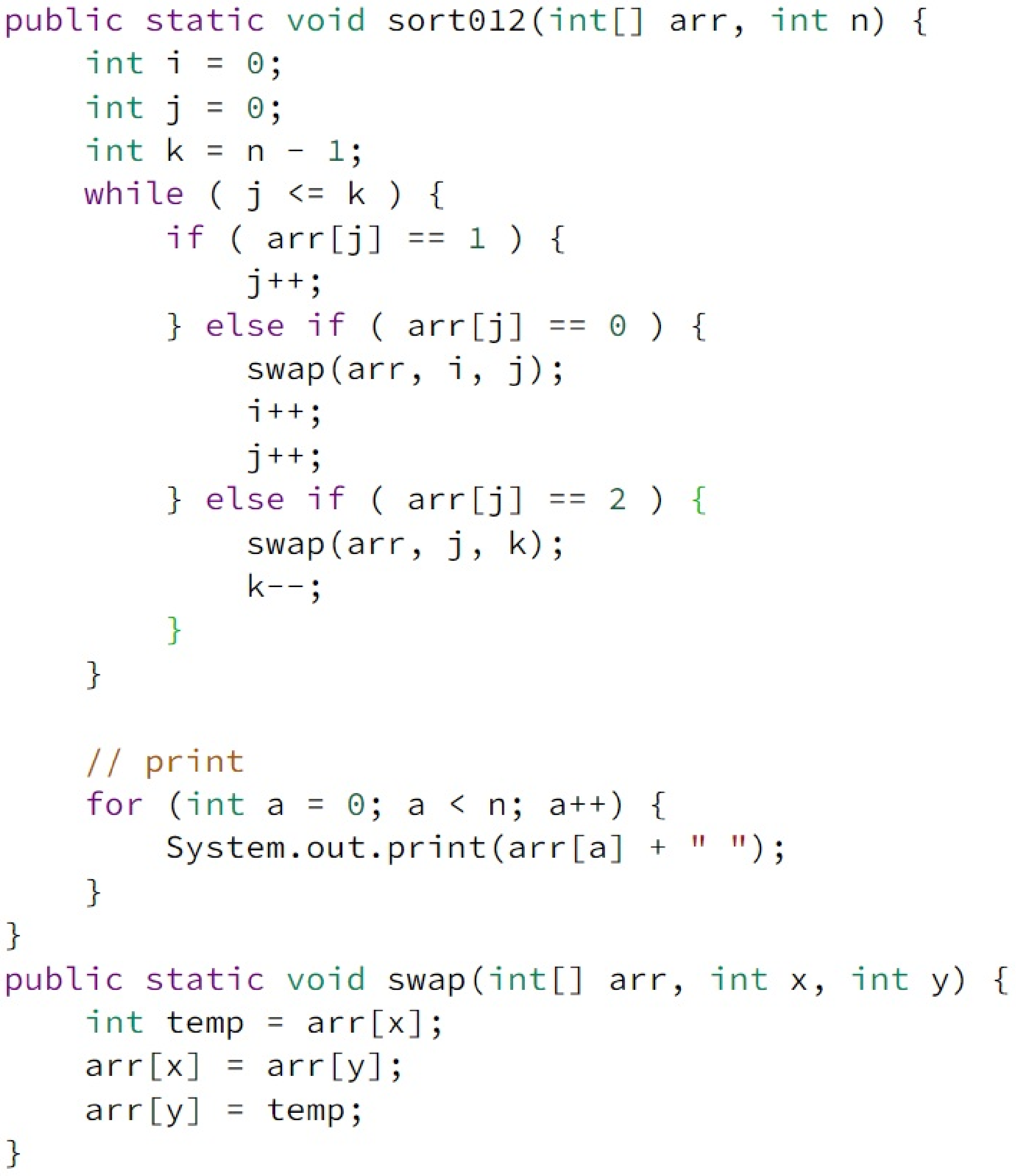
dry run



arr =

	0	1	2	3	4	5	6	7	8	9	10	11
arr =	0	0	0	0	1	1	1	1	1	2	2	2
					↑ i				↑ K	↑ j		

Note :- when swap j and K index then only move K--



```java
public static void sort012(int[] arr, int n) {
    int i = 0;
    int j = 0;
    int k = n - 1;
    while ( j <= k ) {
        if ( arr[j] == 1 ) {
            j++;
        } else if ( arr[j] == 0 ) {
            swap(arr, i, j);
            i++;
            j++;
        } else if ( arr[j] == 2 ) {
            swap(arr, j, k);
            k--;
        }
    }

    // print
    for (int a = 0; a < n; a++) {
        System.out.print(arr[a] + " ");
    }
}
public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

⇒ Variation of 2 pointers

Reach Target

$n = 6$

$$arr = \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ [\,-1, & 1, & 2, & 3, & 4, & 5\,] \end{array}$$

$target = 4$

ans = 0, 5 index
1, 3

$$arr[i] + arr[j] == target$$

$$\begin{array}{cccccccc} & & 0 & 1 & 2 & 3 & 4 & 5 \\ arr & = & [\,-1, & 1, & 2, & 3, & 4, & 5\,] \\ & & \uparrow & & & & & \uparrow \\ & & i & & & & & j \end{array}$$

<u>target = 4</u>

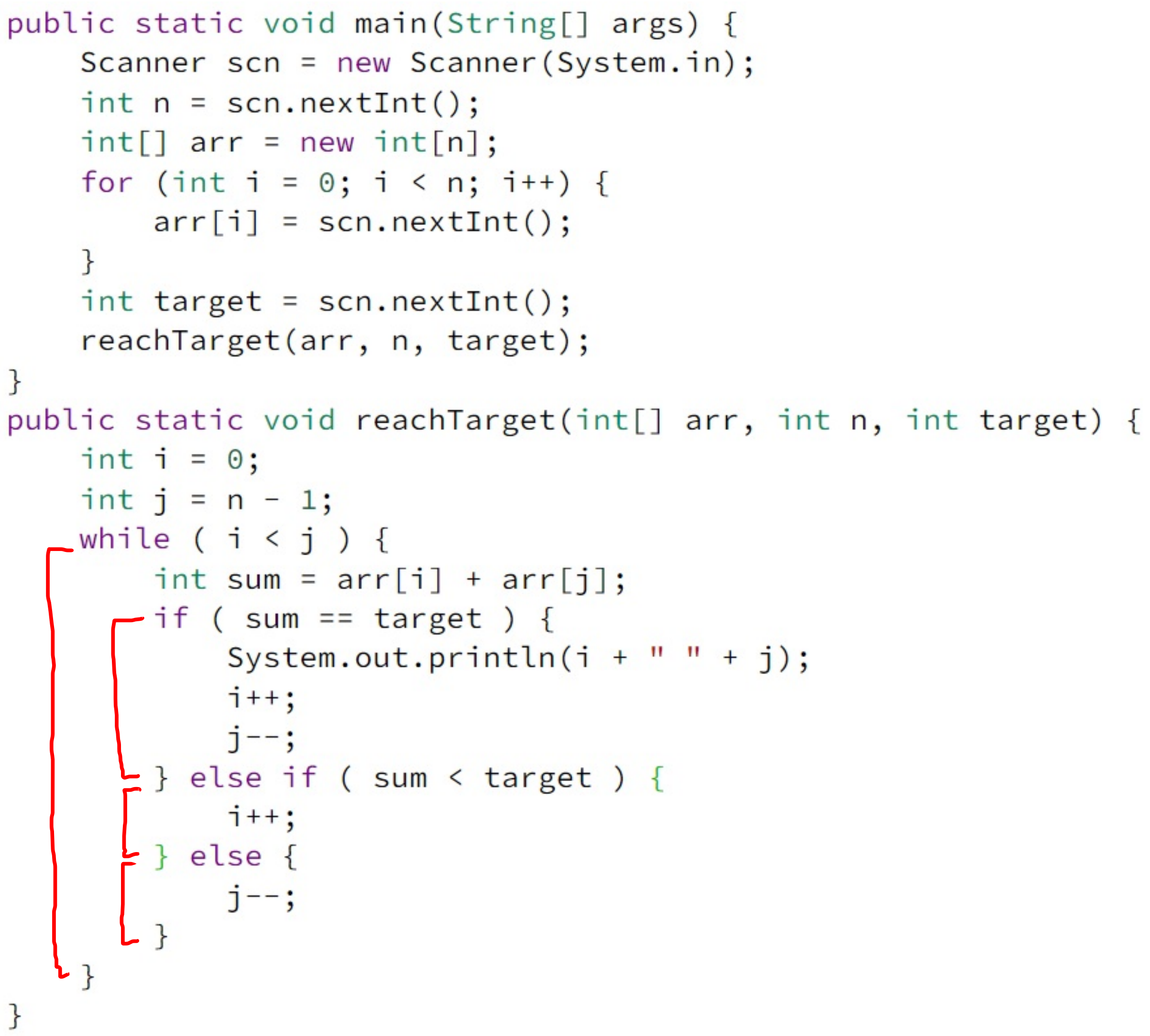
```
sum = arr[i] + arr[j] ;
if ( sum == target)
      Syso ( i + "    " + j );
      i++ , j-- ;
} else if ( sum > target){
      j-- ;
} else if ( sum < target){
      i++ ;
}
```

sum = ~~4~~ ~~5~~ 4

0	5
1	3

Note:- approch will only work if array is sorted

code



```
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();
    }
    int target = scn.nextInt();
    reachTarget(arr, n, target);
}
public static void reachTarget(int[] arr, int n, int target) {
    int i = 0;
    int j = n - 1;
    while ( i < j ) {
        int sum = arr[i] + arr[j];
        if ( sum == target ) {
            System.out.println(i + " " + j);
            i++;
            j--;
        } else if ( sum < target ) {
            i++;
        } else {
            j--;
        }
    }
}
```

Target Sum

(very imp)

- array is not sorted
- may contain duplicates
- only print unique pair

arr =

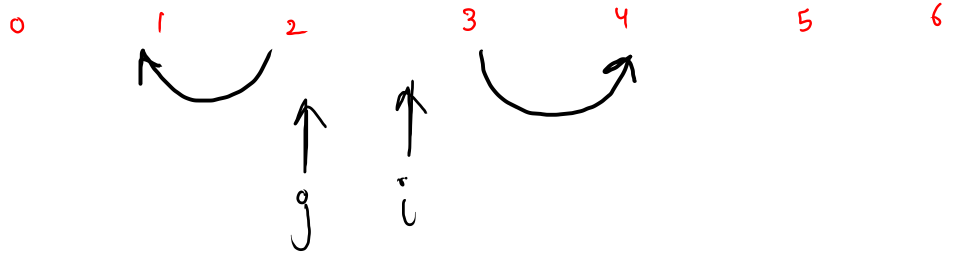
0	1	2	3
2	3	3	4
	↑ j	↑ i	

target = 6

o/p :- 2 , 4

3 , 3

arr = [1 3 3 5 5 7 8], target = 8



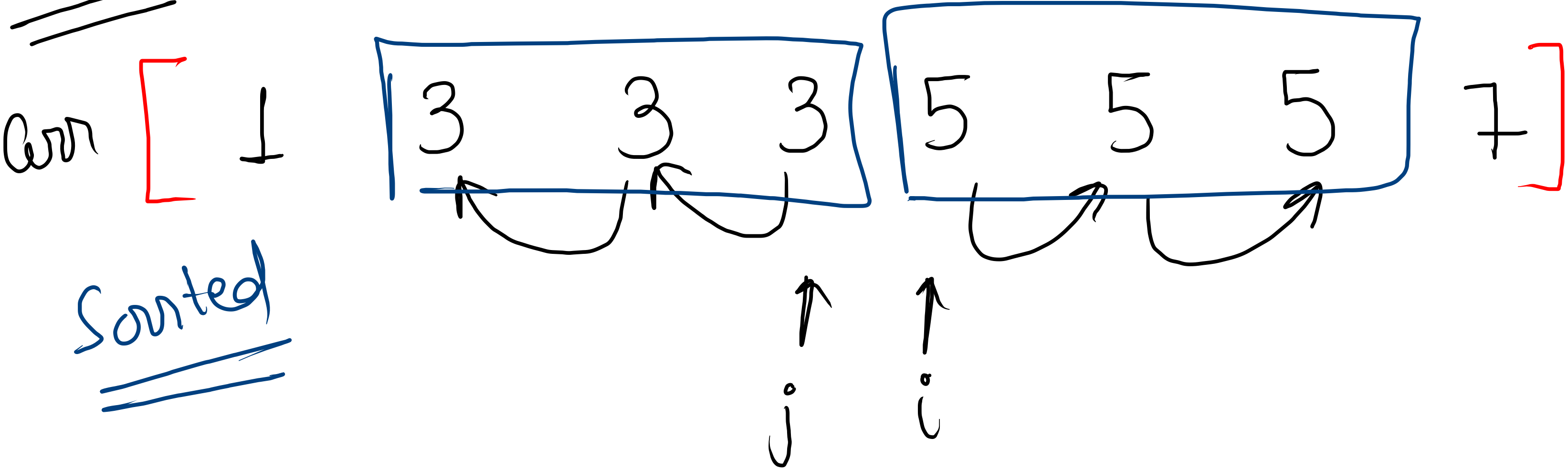
psudo code

```
1) Sort the array
2) declare i = 0 , j = n-1;
3) loop until i<j
    3.1) sum = arr [i] + arr [j]
    3.2) sum < target
            i++
    3.3) sum > target
            j--
    3.4) sum == target
            Syso ( arr[i] + "  " + arr[j]);
            i++
            j--
            while ( arr[i] == arr[i-1]){
                i++;
            }
            while ( arr [j] == arr [j+1] ){
                j--;
            }
```

1	7
3	5

Case

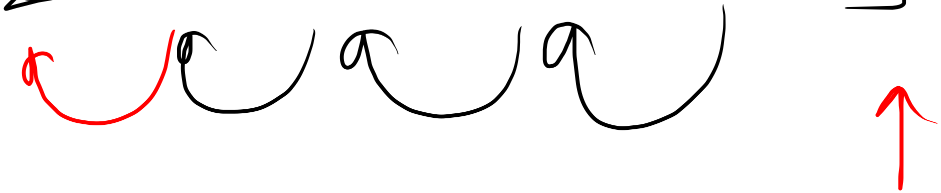
target = 8



1, 7
3, 5

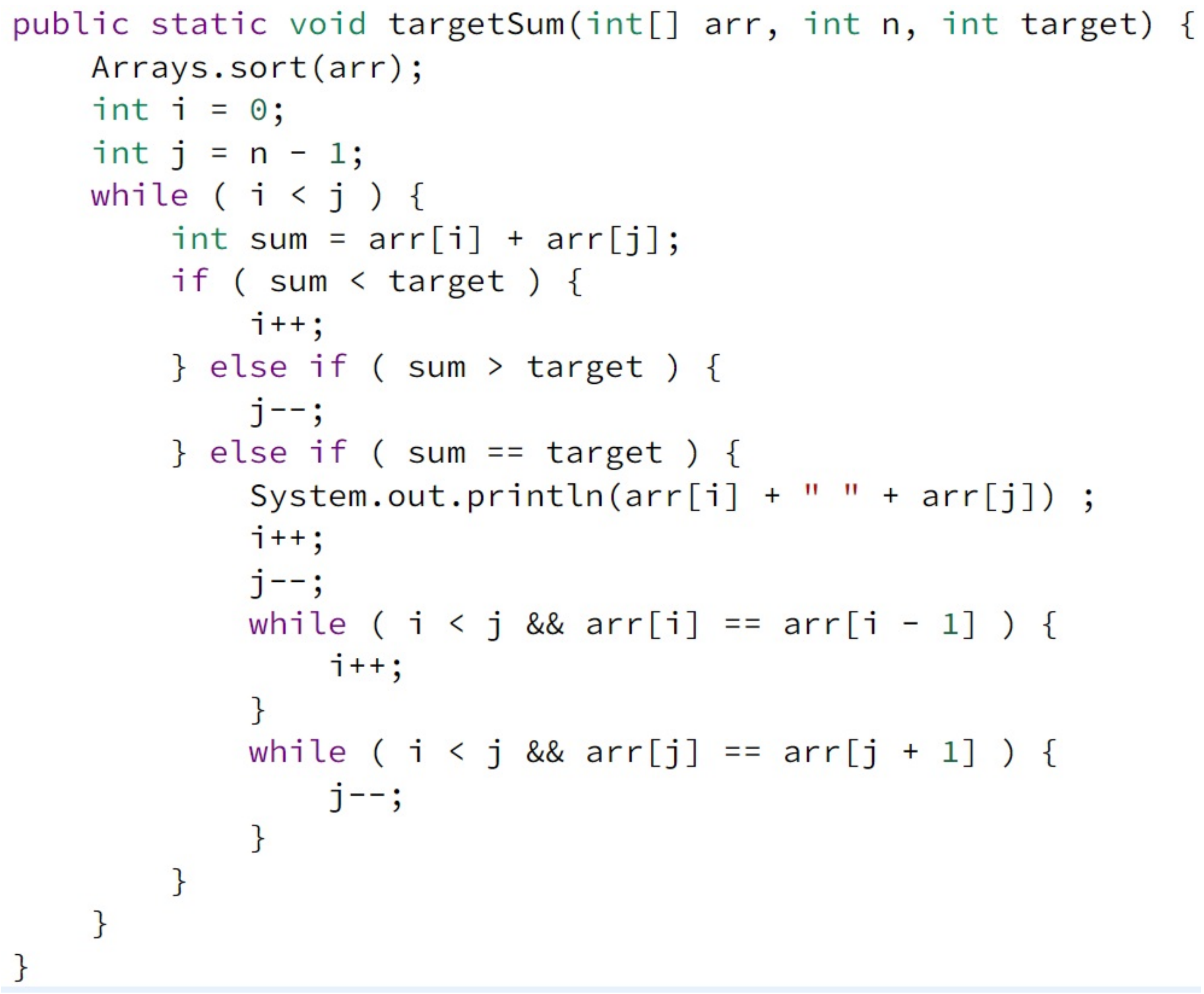
Ex:-

$[1 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2]$



$t = 10$

Code



```java
public static void targetSum(int[] arr, int n, int target) {
    Arrays.sort(arr);
    int i = 0;
    int j = n - 1;
    while ( i < j ) {
        int sum = arr[i] + arr[j];
        if ( sum < target ) {
            i++;
        } else if ( sum > target ) {
            j--;
        } else if ( sum == target ) {
            System.out.println(arr[i] + " " + arr[j]) ;
            i++;
            j--;
            while ( i < j && arr[i] == arr[i - 1] ) {
                i++;
            }
            while ( i < j && arr[j] == arr[j + 1] ) {
                j--;
            }
        }
    }
}
```