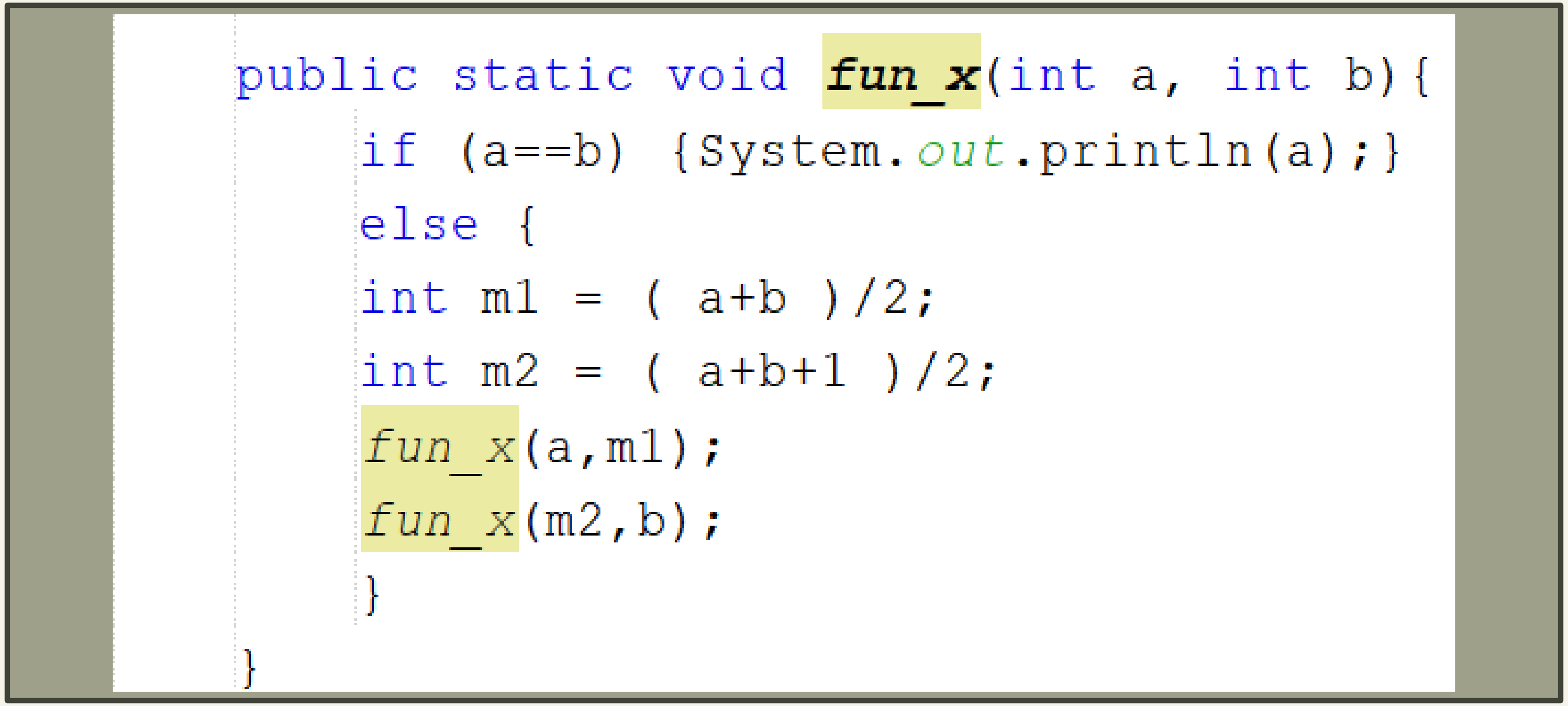
Recursion

sec 3

Tracing recursive codes

What is the output of fun_x(0,8) ?



```
public static void fun_x(int a, int b){
        if (a==b) {System.out.println(a);}
        else {
        int m1 = ( a+b )/2;
        int m2 = ( a+b+1 )/2;
        fun_x(a,m1);
        fun_x(m2,b);
        }
}
```

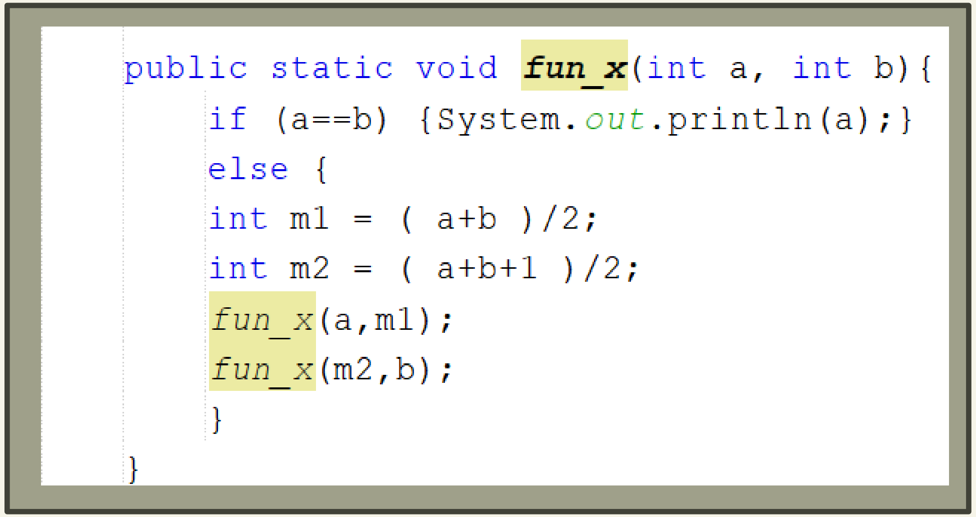
What is the output of fun_x(0,8) ?

fun_x(0,8)=
m1=4
m2=4
fun_x(0,4)
fun_x(4,8)

fun_x(0,4)=
m1=2
m2=2
fun_x(0,2)
fun_x(2,4)

fun_x(0,2)=
m1=1
m2=1
fun_x(0,1)
fun_x(1,2)

fun_x(0,1)=
m1=0
m2=1
0 ← fun_x(0,0)
1 ← fun_x(1,1)

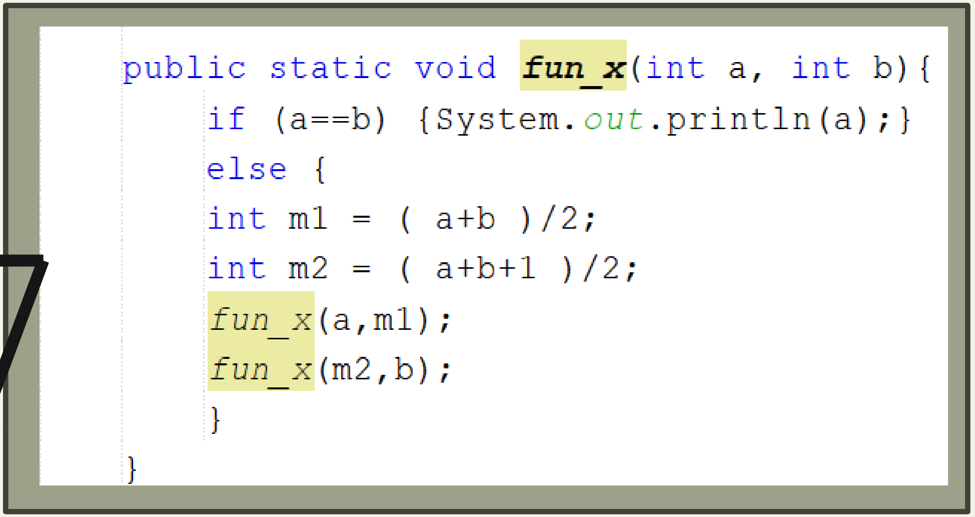


```
public static void fun_x(int a, int b){
        if (a==b) {System.out.println(a);}
        else {
        int m1 = ( a+b )/2;
        int m2 = ( a+b+1 )/2;
        fun_x(a,m1);
        fun_x(m2,b);
        }
}
```

What is the output of fun_x(0,8) ?

fun_x(1,2)=
m1=1
m2=2
fun_x(1,1) → 1
fun_x(2,2) → 2

fun_x(2,4)=
m1=3
m2=3
fun_x(2,3)
fun_x(3,4)

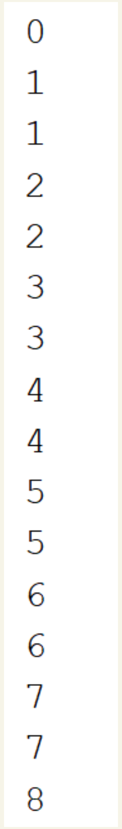


```
public static void fun_x(int a, int b){
     if (a==b) {System.out.println(a);}
     else {
     int m1 = ( a+b )/2;
     int m2 = ( a+b+1 )/2;
     fun_x(a,m1);
     fun_x(m2,b);
     }
}
```

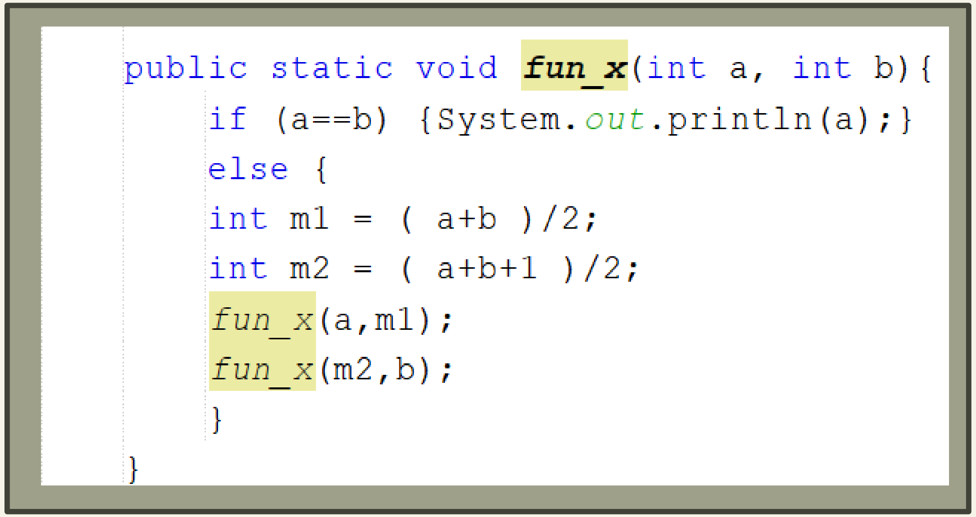
fun_x(2,3)=
m1=2
m2=3
fun_x(2,2) → 2
fun_x(3,3) → 3

fun_x(3,4)=
.
.
.
.

What is the output of fun_x(0,8) ?



```
0
1
1
2
2
3
3
4
4
5
5
6
6
7
7
8
```



```
public static void fun_x(int a, int b){
     if (a==b) {System.out.println(a);}
     else {
     int m1 = ( a+b )/2;
     int m2 = ( a+b+1 )/2;
     fun_x(a,m1);
     fun_x(m2,b);
     }
}
```

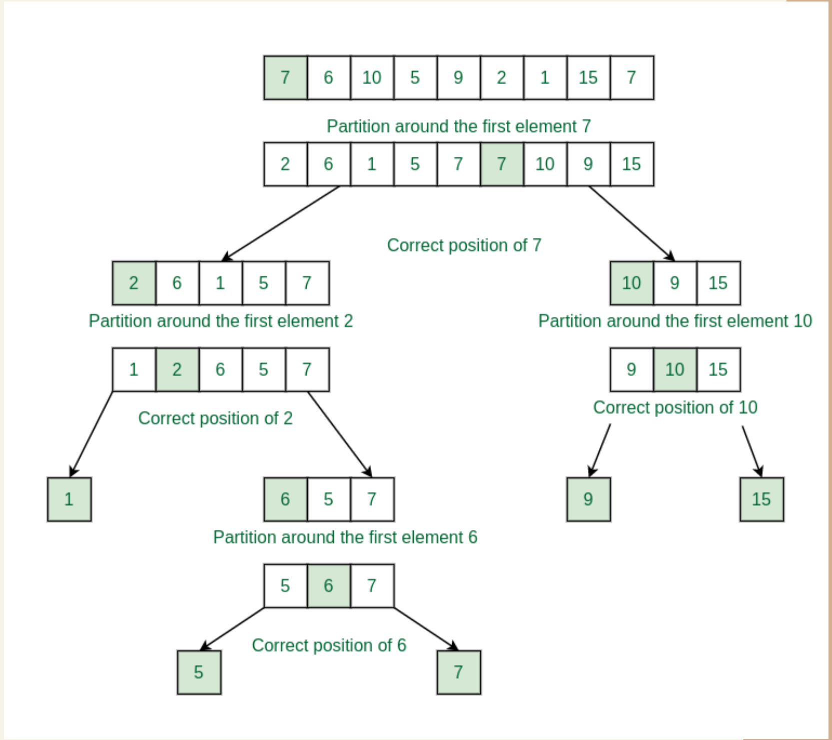
Codes

Quick Sort

Divide-and-conquer based algorithm

•Choose a Pivot: Select an element from the array as the pivot. The choice of pivot can vary (e.g., first element, last element, random element, or median).

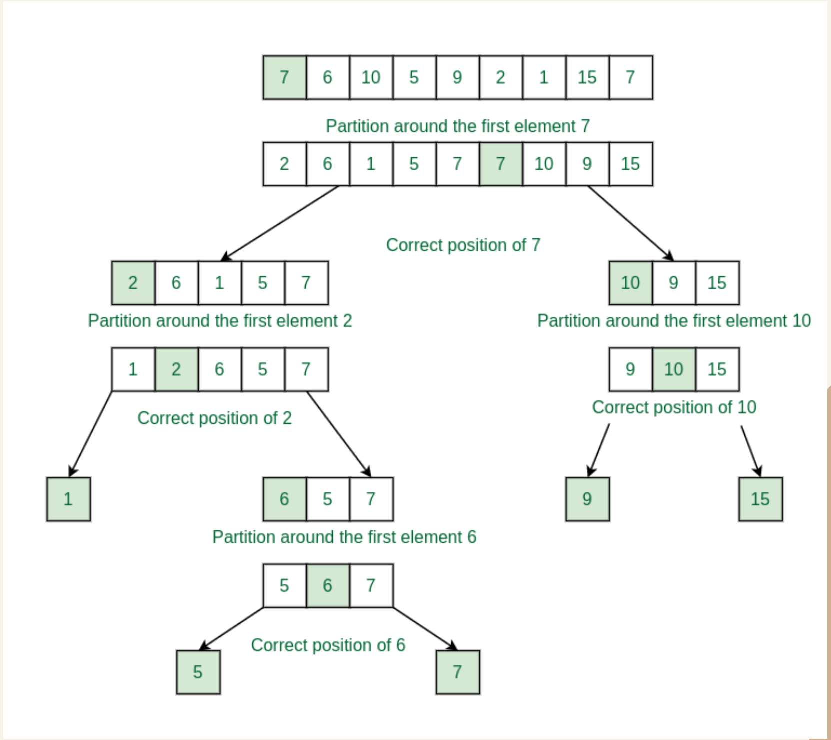
•Partition the Array: Rearrange the array around the pivot. After partitioning, all elements smaller than the pivot will be on its left, and all elements greater than the pivot will be on its right.



Quick Sort

•Recursively Call: Recursively apply the same process to the two partitioned sub-arrays (left and right of the pivot).

•Base Case: The recursion stops when there is only one element left in the sub-array, as a single element is already sorted.



Step1

Pivot = 4

4	8	7	5	1	9	6	3

left (i)

right (j)

$4 < 4$ X

$3 > 4$ X

swap

3	8	7	5	1	9	6	4

left (i)

right (j)

$3 < 4$

i++

3	8	7	5	1	9	6	4

left (i)

right (j)

$8 < 4$ X

$4 > 4$ X

swap

Step1

Pivot = 4

3	4	7	5	1	9	6	8
	left (i)						right (j)

$4 < 4$ X

$8 > 4$

j --

3	4	7	5	1	9	6	8
	left (i)					right (j)	

$6 > 4$

j --

3	4	7	5	1	9	6	8
	left (i)				right (j)		

$9 > 4$

j --

Step1

Pivot = 4

3	4	7	5	1	9	6	8
	left (i)			right (j)			

1 > 4 X

swap

3	1	7	5	4	9	6	8
	left (i)			right (j)			

1 < 4

i ++

3	1	7	5	4	9	6	8
		left (i)		right (j)			

7 < 4 X

4 > 4 X

swap

Step1

Pivot = 4

3	1	4	5	7	9	6	8
		left (i)		right (j)			

4 < 4 X

7 > 4

j --

3	1	4	5	7	9	6	8
		left (i)	right (j)				

5 > 4

j --

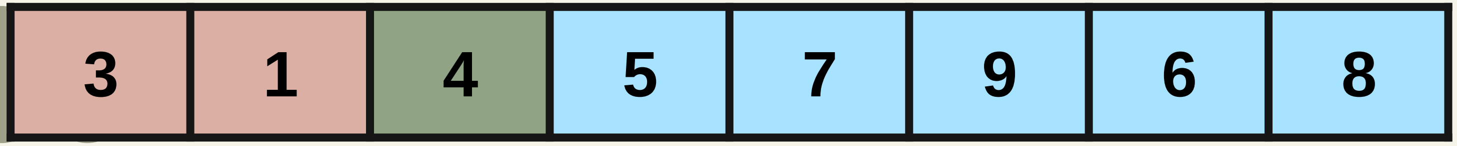
3	1	4	5	7	9	6	8
		left (i) right (j)					

4 < 4 X

4 > 4 X

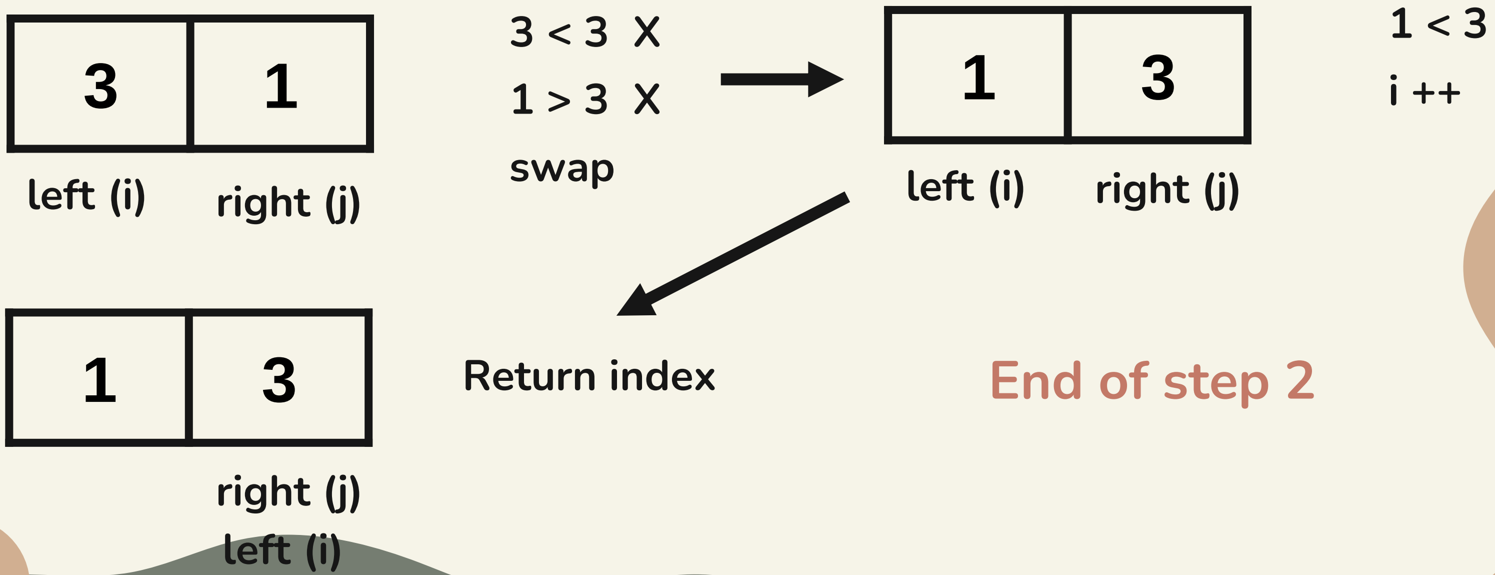
return index

End of step 1

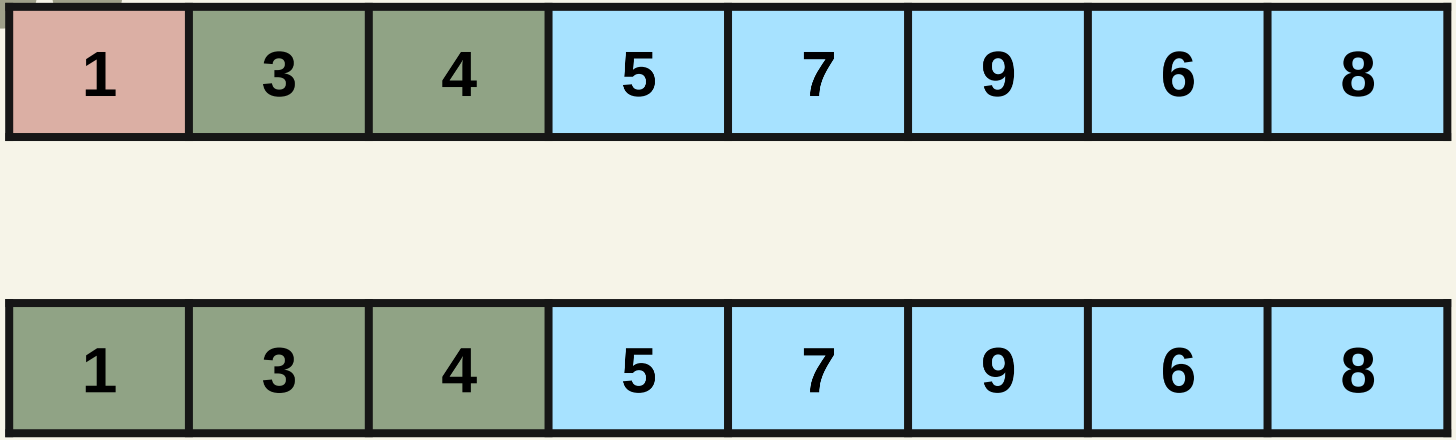


Step2

Pivot = 3

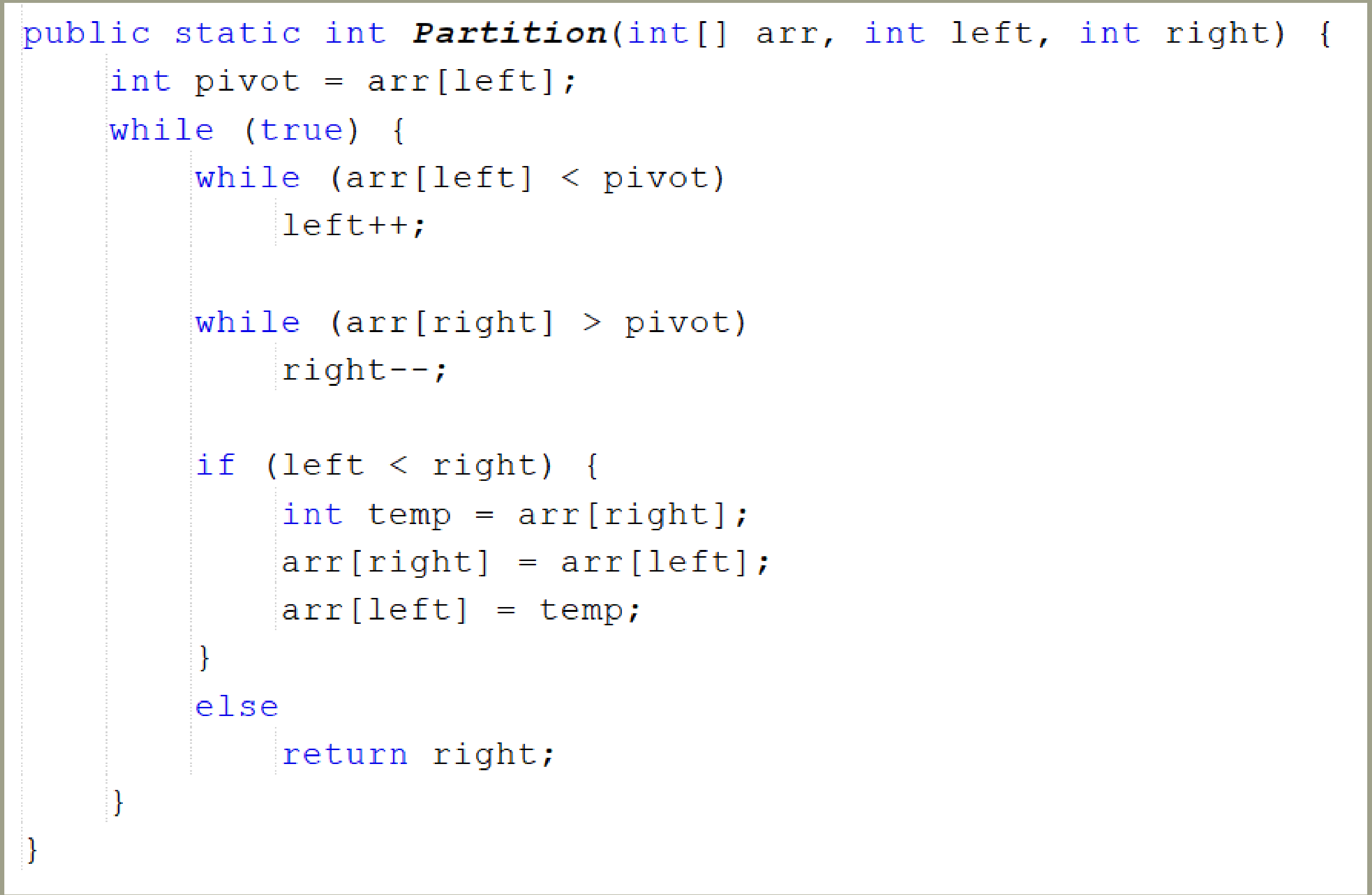


End of step 2



Try to complete the sort

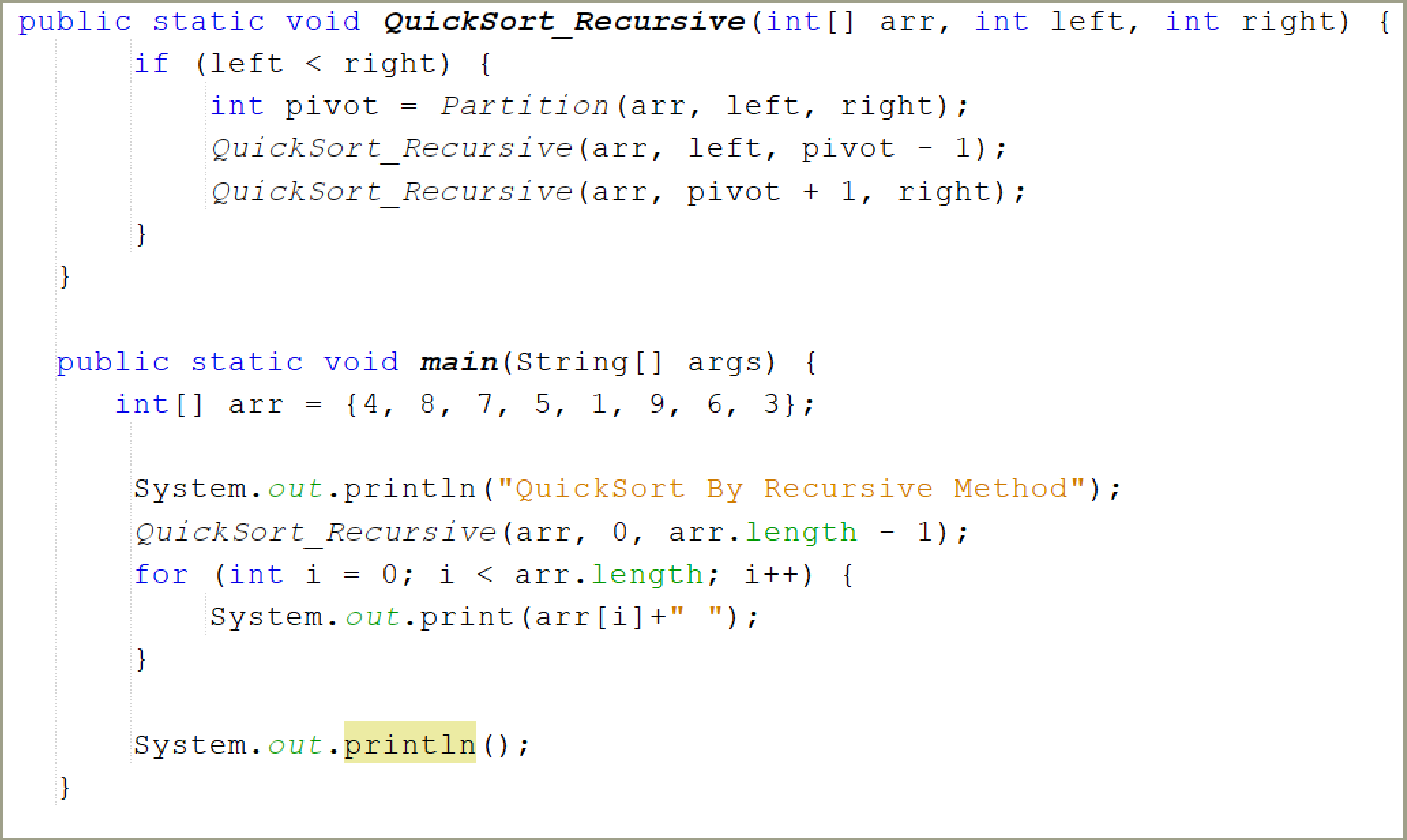
How many steps ?



```java
public static int Partition(int[] arr, int left, int right) {
    int pivot = arr[left];
    while (true) {
        while (arr[left] < pivot)
            left++;

        while (arr[right] > pivot)
            right--;

        if (left < right) {
            int temp = arr[right];
            arr[right] = arr[left];
            arr[left] = temp;
        }
        else
            return right;
    }
}
```



```
public static void QuickSort_Recursive(int[] arr, int left, int right) {
      if (left < right) {
          int pivot = Partition(arr, left, right);
          QuickSort_Recursive(arr, left, pivot - 1);
          QuickSort_Recursive(arr, pivot + 1, right);
      }
  }

  public static void main(String[] args) {
     int[] arr = {4, 8, 7, 5, 1, 9, 6, 3};

      System.out.println("QuickSort By Recursive Method");
      QuickSort_Recursive(arr, 0, arr.length - 1);
      for (int i = 0; i < arr.length; i++) {
          System.out.print(arr[i]+" ");
      }

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

Thanks ☻

Eng / Hajar Nagm