```
int partition(int [ ] array, int size, int pivot){
        int down=0, up=size-1;
        while(down<up){
                // finds the first, from left, element that is greater than pivot
                while(down<=up && array [down]<=pivot){
                        down++;
                }

                // finds the first, from right, element that is less than or equal to pivot
                while(up>=down && array [up]>pivot){
                        up--;
                }

                if(down<up){
                        // exchange array[down] and array[up]
                        int temp= array [down];
                        array [down]= array [up];
                        array [up]=temp;
                }
        }
        return up;
}
```

Example 1

Given a pivot **30** and an array as follows:

Array elements	51	3	34	6	5	1	31	19	72	48
Index	0	1	2	3	4	5	6	7	8	9

See how ***down***, ***up***, and ***array*** are changing during the execution:

At the end of the 1st iteration of the outer while loop:

down	up	array[down]	array[up]
0	7	51 (>pivot)	19 (<=pivot)

19 ←→51

Array elements	**19**	3	34	6	5	1	31	**51**	72	48
Index	0	1	2	3	4	5	6	7	8	9

At the end of the 2nd iteration of the outer while loop:

down	up	array[down]	array[up]
2	5	34 (>pivot)	1 (<=pivot)

1 ←→34

Array elements	19	3	**1**	6	5	**34**	31	51	72	48
Index	0	1	2	3	4	5	6	7	8	9

At the end of the 3rd iteration of the outer while loop:

down	up	array[down]	array[up]
5	4	34 (>pivot)	5 (<=pivot)

No exchange because down>=up. The outer while loop ends and **4** is returned.

Array elements at 0, 1, 2, 3, and 4 are now all less than or equal to the pivot **30**. Array elements at 5, 6, 7, 8, and 9 are all greater than the pivot **30**.