5) Show the result of running Partition (as shown in class on Friday) on the array below using the leftmost element as the pivot element. Show what the array looks like after each swap.

Initial	5	2	1	7	8	3	4	6
	5	2	1	4	8	3	7	6
	5	2	1	4	3	8	7	6
After Partition	3	2	1	4	5	8	7	6

6) Show the contents of the array below after each merge occurs in the process of Merge-Sorting the array below:

Initial	3	6	8	1	7	4	5	2
	3	6	8	1	7	4	5	2
	3	6	1	8	7	4	5	2
	1	3	6	8	7	4	5	2
	1	3	6	8	4	7	5	2
	1	3	6	8	4	7	2	5
	1	3	6	8	2	4	5	7
Last	1	2	3	4	5	6	7	8

7) Here is the code for the partition function (used by Quick Sort). Explain the purpose of each line of code.

```
int partition(int* vals, int low, int high) {

  int lowpos = low; // set the pivot
  low++; //increment to after the pivot

  while (low <= high) { // loop until low and high cross
    //increment if from left each val is ≤ the pivot
    while (low <= high && vals[low] <= vals[lowpos]) low++;
    while (high >= low && vals[high] > vals[lowpos]) high--;
    //decrement if from right each val is > pivot
    if (low < high)// if low and high are not yet crossed
      swap(&vals[low], &vals[high]);//swap the values of low and high
  }// continue until loop ends

  swap(&vals[lowpos], &vals[high]); //swap the pivot with high (RIGHT) index
  return high;//return the partition index
}
```