Computer Science I – Summer 2011
Recitation #9: Sorting (Solutions)

1) Show the contents of the array below being sorted using Insertion Sort at the end of each loop iteration.

Initial	2	8	3	6	5	1	4	7
	2	**8**	**3**	**6**	**5**	**1**	**4**	**7**
	2	**3**	**8**	**6**	**5**	**1**	**4**	**7**
	2	**3**	**6**	**8**	**5**	**1**	**4**	**7**
	2	**3**	**5**	**6**	**8**	**1**	**4**	**7**
	1	**2**	**3**	**5**	**6**	**8**	**4**	**7**
	1	**2**	**3**	**4**	**5**	**6**	**8**	**7**
Sorted	1	2	3	4	5	6	7	8

2) Show the contents of the array below being sorted using Selection Sort at the end of each loop iteration. As shown in class, please run the algorithm by placing the smallest item in place first.

Initial	6	2	8	1	3	7	5	4
	1	**2**	**8**	**6**	**3**	**7**	**5**	**4**
	1	**2**	**8**	**6**	**3**	**7**	**5**	**4**
	1	**2**	**3**	**6**	**8**	**7**	**5**	**4**
	1	**2**	**3**	**4**	**8**	**7**	**5**	**6**
	1	**2**	**3**	**4**	**5**	**7**	**8**	**6**
	1	**2**	**3**	**4**	**5**	**6**	**8**	**7**
Sorted	1	2	3	4	5	6	7	8

3) Show the contents of the array below being sorted using Bubble Sort at the end of each loop iteration. As shown in class, please run the algorithm by placing the largest item in place first.

Initial	4	2	6	5	7	1	8	3
	2	**4**	**5**	**6**	**1**	**7**	**3**	**8**
	2	**4**	**5**	**1**	**6**	**3**	**7**	**8**
	2	**4**	**1**	**5**	**3**	**6**	**7**	**8**
	2	**1**	**4**	**3**	**5**	**6**	**7**	**8**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Sorted	1	2	3	4	5	6	7	8

4) When Merge Sort is run on an array of size 8, the merge function gets called 7 times. Consider running Merge Sort on the array below. What would the contents of the array be right before the 7^{th} call to the Merge function?

Initial	7	2	1	5	8	3	4	6
Before 7^{th} Merge	**1**	**2**	**5**	**7**	**3**	**4**	**6**	**8**

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; // Stores lowest index in range.
  low++; // Starts low pointer at 2nd value.

  while (low <= high) { // Loops until low and high cross over

    // Find next item on left that is too big.
    while (low <= high && vals[low] <= vals[lowpos]) low++;

    // Find next item on right that is too small.
    while (high >= low && vals[high] > vals[lowpos]) high--;

    // If these items are out of place, swap them.
    if (low < high)
      swap(&vals[low], &vals[high]);
  }

  // Swap the partition value into its correct location.
  swap(&vals[lowpos], &vals[high]);
  return high; // This is index storing the partition element.
}
```