Data Structures (810:052) Lab 8 Name:___________________

Objectives: You will gain experience:

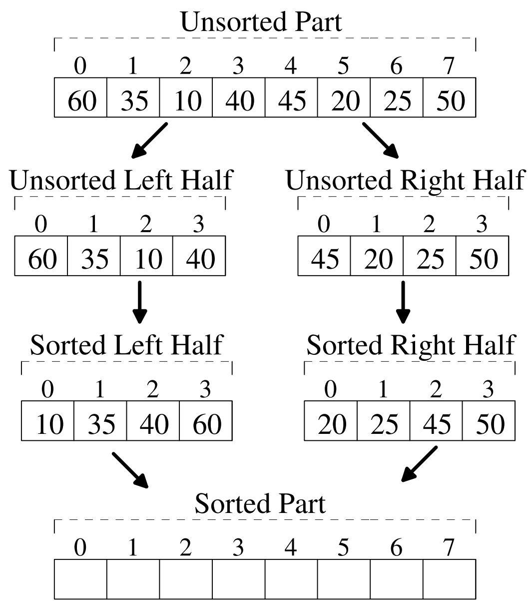
- get a feel for advanced sorts: heap, quick, and merge sorts
- write some recursive list functions

Download the following file to your desktop: http://www.cs.uni.edu/~fienup/cs052s10/labs/lab8.zip
Extract this file to the Desktop by right-clicking on lab6.zip icon and selecting Extract All.

Part A: The `lab8.zip` file you downloaded and extracted contains a `advancedSorts` folder with a Visual Studio C++ project file: `binaryHeap.sln` inside. Double-click on it to open this project in Visual Studio. Your task is to implement the recursive `mergeSort` function which we discussed in class yesterday. The `main.cpp` file contains code to test and time your sort functions.

The general idea merge sort is as follows. Assume "size" items to sort.

- Split the unsorted part in half to get two smaller sorting problems of about size/2
- Solve both smaller problem recursively using merge sort
- "Merge" the solution to the smaller problems together to solve the original sorting problem of size n (the `merge` function is provided)



After you have implemented and tested you mergeSort function, raise your hand and demonstrate your program.

Part B: *Quick sort* is another advanced sort that often is quicker than merge sort (hence its name). The general idea is as follows. Assume "n" items to sort.

- Select a "random" item in the unsorted part as the *pivot*
- Rearrange (called *partitioning*) the unsorted items such that:

Pivot Index

All items < to Pivot	Pivot Item	All items >= to Pivot

- Quick sort the unsorted part to the left of the pivot
- Quick sort the unsorted part to the right of the pivot

In the `main.cpp` file you are given the following `partition` function which returns the index of the pivot after this rearrangement. Your task is to write the recursive quickSort function

```
int partition(int set[], int start, int end) {
   int pivotValue, pivotIndex, mid;

   mid = (start + end) / 2;
   swap(set[start], set[mid]);
   pivotIndex = start;
   pivotValue = set[start];
   for (int scan = start + 1; scan <= end; scan++) {
      if (set[scan] < pivotValue) {
         pivotIndex++;
         swap(set[pivotIndex], set[scan]);
      } // end if
   }// end for
   swap(set[start], set[pivotIndex]);
   return pivotIndex;
} // end partition

// swap simply exchanges the contents of value1 and value2.
void swap(int &value1, int &value2) {
   int temp = value1;
   value1 = value2;
   value2 = temp;
} // end swap
```

After you have implemented and tested you `quickSort` function, raise your hand and demonstrate your program.

Part C: Comment out the printing of the sorted arrays, then run the current main program (in `main.cpp`) to complete the following timings.

Sort Algorithm	Sort Time of 2,000,000 Elements (seconds)
heap sort	
merge sort	
quick sort	

a) All three of these algorithms are $\Theta(n\log_2 n)$ on initially random data. Why do you suppose merge sort is the slowest?

After you have completed the above times and answered the above question, raise your hand and explain your answers.