For Problem 1, consider the following C++ function:

```cpp
void mysterySort(T data[], int n)
{
    for (i = 0; i < n-1; i++)
    {
        for (j = n-1; j > i; --j)
        {
            if (data[j] < data[j-1])
            {
                T tmp = data[j];
                data[j] = data[j-1];
                data[j-1] = tmp;
            }
        }
    }
}
```

For Problems 2 – 6, consider Tree T shown below:

```
Tree T
    40
   / \
 24   55
 / \
11   31
  / \
27   47
     / \
    35
```
1. [15 points] Fill-in the table below to indicate in Big-$\Theta$ notation the number of comparisons and the number of moves that $\text{mysterySort}$ will make in the best, worst, and average case. **In order to receive full credit, in the space after the table, you must fully justify your answer for each cell in the table.**

<table>
<thead>
<tr>
<th></th>
<th>Best-case</th>
<th>Worst-Case</th>
<th>Average-Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Comparisons</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Moves</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. [5 points] Starting with Tree T as shown on the first page, draw the AVL tree that would result from inserting 5 and 18 (in that order) into Tree T.

3. [5 points] Starting with Tree T as shown on the first page, draw the AVL tree that would result from inserting 5 and 3 (in that order) into Tree T.
4. [5 points] Starting with Tree T as shown on the first page, draw the AVL tree that would result from inserting 5, 18, and 3 (in that order) into Tree T.

5. [5 points] Starting with Tree T as shown on the first page, draw the AVL tree that would result from deleting 27 and 35 (in that order) from Tree T.
6. [5 points] Starting with Tree T as shown on the first page, draw the AVL tree that would result from deleting 11 from Tree T.

7. [10 points] Consider calling the BuildHeap function shown on slide 4 of the class notes on Heaps with the following array as input: [36, 24, 21, 2, 18, 3, 7, 12, 10, 41, 25]. Show the array at the end of each iteration of the for loop in the BuildHeap function. (Hint: It may help to draw the heap on scratch paper).

<table>
<thead>
<tr>
<th>Array[]</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>36</td>
<td>24</td>
<td>21</td>
<td>2</td>
<td>18</td>
<td>3</td>
<td>7</td>
<td>12</td>
<td>10</td>
<td>41</td>
<td>25</td>
</tr>
<tr>
<td>Idx = 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idx = 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idx = 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idx = 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idx = 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8. [10 points] Using the implementation of quicksort shown on slides 40 and 41 of the class notes on Sorting, illustrate the operation of quicksort on the array [36, 24, 21, 2, 18, 3, 7, 12, 10, 41, 25]. Note that this implementation of quicksort chooses the middle-most element as the pivot value. To illustrate the operation of quicksort on the array, show the array at the end of each call to Partition. You may not need to use all the rows in the table provided below, show as many rows in the table below as are needed.

<table>
<thead>
<tr>
<th>Pivot</th>
<th>LastPreceder</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>start</td>
<td>start</td>
<td>36</td>
<td>24</td>
<td>21</td>
<td>2</td>
<td>18</td>
<td>3</td>
<td>7</td>
<td>12</td>
<td>10</td>
<td>41</td>
<td>25</td>
</tr>
</tbody>
</table>