These must be completed and shown to your lab TA either by the end of this lab, or at the start of your next lab. You may work in groups of up to two people.

1. Download `binaryHeap.cpp` from the course web page under Lab 4. You need to complete the `printHeap` function so that it prints the contents of the heap in a "tree-like" fashion. For example, if your heap (as an array) is `[0,2,1,4,3,9,5,7,6,8]`, then `printHeap` should output:

```
0
*2
**4
***7
***6
**3
***8
*1
**9
**5
```

   First print the current element, then its left subtree, and then its right subtree. Preface each element with a number of asterisks equal to its depth in the heap.

2. Once you are done with the previous exercise, rewrite the `printHeap` function in `binaryHeap.cpp` to print the heap in a different tree-like format:

```
        5
    1
        9
0
        3
            8
    2
            6
        4
            7
```

   If you rotate this output by 90 degrees, you can see the tree. Can you modify the code you wrote for Question 1 to generate this result?

   The remaining questions are designed to emphasize the fact that often the **simplest** algorithm is best.

3. Implement the following function:

```
//PRE:  heap points to an array representing a heap
//      key is the value to be removed from the heap
//      size is the number of elements in the heap
//POST: all elements with key value = key have been removed from
// the heap and size is the new heap size.
void remove(int* heap, int key, int & size);
```

   Are the `remove` tests printing out correctly? Be sure to draw a picture and refer to your notes if you're unsure what to expect.

4. Implement the following function:

   ```
   //PRE:  heap1 and heap2 contain size1 and size2 elements respectively.
   //POST: output a new heap (whose size is size1+size2) containing all
   // the elements in heap1 and heap2 (including duplicates).
   int* mergeHeap(int* heap1, int* heap2, int size1, int size2);
   ```

   Is the `merge` test printing out correctly? Be sure to draw a picture and refer to your notes if you're unsure what to expect.

5. What is the asymptotic running time of your solution to `remove` (as a function of `size`)? What is the asymptotic running time of your solution to `merge` (as a function of `size1` and `size2`)? Be prepared to explain how your code works.

6. Be sure to show your work to your TA before you leave, or at the start of the next lab, or you will not receive credit for the lab!