CSE 101
Midterm 2 Review Problems

1. Rank the following functions from lowest to highest asymptotic growth rate.

   1) 2^n
   2) $n \ln(n)$
   3) n
   4) $2^{\ln(n)}$
   5) ln(ln(n))
   6) $n\sqrt{n}$
   7) n^2
   8) $\ln(n^2)$
   9) $\sqrt{n}$

   Write your answer as a permutation of the set $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, giving the corresponding line numbers of the above functions in the required order (left to right, slowest growing function to fastest growing function.) No justifications are required.

2. Consider the List ADT from pa5 but *without* the `cleanup()` function. Write a C++ client function with heading

   ```
   void RemoveDuplicates(List& L)
   ```

   that does the same thing as `cleanup()`, except that it does not matter where the cursor ends up. In other words, the call `RemoveDuplicates(L)` will alter List L so that it contains only the first occurrence of each of its data items. To do this, you may use all ADT operations in List.h *except* `cleanup()`.

3. Let T be a Binary Search Tree containing the keys $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$. Suppose that a **pre-order tree walk** prints the keys in order: $2, 1, 12, 10, 8, 7, 5, 4, 3, 6, 9, 11, 13$, and that a post-order tree walk prints the keys in order: $1, 3, 4, 6, 5, 7, 9, 8, 11, 10, 13, 12, 2$. Determine the structure of T. (Note: only one of the two tree walks is really necessary since each of them uniquely determines the structure of T.) Present your solution either by drawing a picture of the tree, or by constructing a table giving the parent of each Node.

4. Use the `TreeInsert()` algorithm to insert the following keys: 6, 2, 1, 4, 10, 8, 7, 9, 12, 11, 14, 13, 15 (in order) into an initially empty BST.

   a. Draw the resulting BST

   b. Use the `Delete()` algorithm to delete the following keys: 8, 6, 13, 14 (in order) from the BST you drew in part (a), then draw the resulting tree.

5. Suppose we alter the List ADT from pa5 by doing

```
typedef char ListElement;
```

at the beginning of `List.h`, making it a list of `char` instead of `int`. Assume a List L consists entirely of parenthesis characters `'('` and `')'`. The List L is called a *Well Formed Formula* (WFF) iff all parentheses can be matched in pairs (open and close). For instance "`(()(()))`" and "`()()(())`" are WFFs, while "`(()()`" and "`(()))`" are not. The empty List is considered to be a WFF. Write a client function with heading

```
bool isWFF(List L)
```

that returns `true` or `false`, according to whether `L` is or is not a WFF. (Hint: search for adjacent matching pairs and delete them. If `L` becomes empty, then return true.)