CS300 Final Review Questions

This is not a complete list of questions and topics, but a good sampling of questions that will help you study for the final. I strongly advise you to work through every single question.

```
typedef struct NODE *NODE_PTR;
typedef struct NODE
        {
          char data;
          NODE_PTR psNext;
        } NODE;
```

```
typedef struct BT_NODE *BT_NODE_PTR
typedef struct BT_NODE
        {
          int data;
          BT_NODE_PTR psLeftChild;
          BT_NODE_PTR psRightChild;
        } BT_NODE;
```

1) The values A, B, C, D are inserted (in the order listed) into a stack maintained as a circular list. Draw a picture of the resulting stack after all elements have been inserted.

2) The values A, B, C, D are inserted (in the order listed) into a queue maintained as a circular list. Draw a picture of the resulting queue after all elements have been inserted.

3) The stack described in 1) is maintained with a single pointer of type NODE_PTR. Write a function sPush that accepts a stack pointer and a data value. Push the value on to the stack.

4) The queue described in 2) is maintained with a single pointer of type NODE_PTR. Write a function qEnqueue that accepts a queue pointer and a data value. Insert the data into the proper place in the queue.

5) Assume we have a new data structure for a stack as follows:

```
typedef struct STACK
        {
          NODE_PTR psTop;
          unsigned int numElements;
        } STACK;
```

Rewrite 3) using the new data structure.

6) Assume we have a new data structure for a queue as follows:

```
typedef struct QUEUE
        {
          NODE_PTR psPtr;
```

```
               unsigned int numElements;
             } QUEUE;
```

Rewrite 4) using the new data structure.

7) Show what a call would look like for the functions described in 3), 4), 5), and 6).

8) What is the computing complexity for determining the size of the stack in questions 3 and 5?

9) What is the computing complexity for determining the size of the queue in questions 4 and 6?

10) Insert the following values into a BST: 40, 30, 35, 60, 80, 70, 32, 25, 27.

11) Insert the following values into an AVL tree: 40, 30, 35, 60, 80, 70, 32, 25, 27.

12) What is the worst-case computing complexity for searching a: a) BST b) AVL tree c) ordered array d) unordered array e) ordered list f) unordered list.

13) What is the worst-case computing complexity for inserting into a: a) BST b) AVL tree c) ordered array d) unordered array e) ordered list f) unordered list.

14) 5) (10 pts) Consider the following C program segment that is intended to insert an item, key, into a singly linked list (pointed to by psList) with ascending keys.

```
ListPtr psTrav, psTemp;

psTrav = psList;
if (NULL != psTrav)
{
  while (psTrav->data < key)
  {
    psTrav = psTrav->psNext;
  }
}

psTemp = (ListPtr) malloc (sizeof (ListNode));
psTemp->data = key;
psTemp->psNext = psTrav->psNext;
psTrav->psNext = psTemp;
```

When the segment is executed with a value for key that does not match any value in the list, but is smaller than some value in the list, what happens?

15) Write a function btCountNodes that returns the number of nodes in a Binary Tree. What does a call to your function look like?

16) Write a function btLargest that returns the largest value in a: a) BST b) BT. What does a call for each function look like?

17) Write a function lstIsEqual that accepts two list pointers of type NODE_PTR and returns TRUE if the two lists are the same; otherwise, FALSE is returned.

18) Review hash tables including: a) hash methods b) collision handling techniques, c) the concepts of primary and secondary clustering

19) What are the advantages of generic programming?

20) Make sure you understand the specifics of pointers, handles, dynamic memory, activation records, the heap.

21) Go over all old exams and make sure you understand everything about the problems you missed points on.