 name _____

CS 112 – Data Structures

Summer, 2012

Midterm Practice Exam I

Please note that this is from 2012. The topic coverage may be different this year, but this exam gives you a good idea of the kind of questions I tend to ask.

1.

a. For each of the following functions give its simplest big-O equivalent.

E.g. for F(n) = 5 the answer would be O(1)

i. $F(n) = n^2 + 100$ is O(_____)

ii. $F(n) = ((n+1)*(n-1))^{0.5}$ is O(_____)

iii. $F(n) = 2^{n*n}$ is O(_____)

iv. $F(n) = n * \log_2(n)$ is O(_____)

2. Suppose you have a sorted linked list of n positive integers and another integer (the target) and you want to find the smallest integer in the list at least as large as the target. For instance, if the list is 2 → 33 → 42 → null (n is 3) and the target is 20, since the list is sorted, as soon as you see the 33 you would stop and return the 33, since 33 >= 20. How many elements of the list would you have to compare with the target? Give a specific answer in terms of n, not in terms of big-O. Also explain how you get this answer.

a. worst case? _____ why?

b. best case? _____ why?

3. Consider the following problem: Given a sorted array S holding s numbers and an unsorted linked list U holding u numbers, find all the numbers in U that are not in S. For instance if S holds 3 6 44 45 and U holds 2 6 3 1 the answer is 2 1. Assume that there are no duplicates within U or within S. Suppose Algorithm A is to go through U element by element and check each one to see if it is in S. What is the big-O worst case complexity of Algorithm A?

4.

a. Suppose you start with an empty stack and do the following operations. For each pop, show the data that would be returned by the pop. If any operation causes an error, say so.

Push 7, push 5 push 4, pop: result _____, pop:result _____, pop: result _____, pop: result _____

b. Do the same as a above but for a queue.

Enqueue 4, enqueue 5, dequeue: result: _____, enqueue 3, dequeue: result_____

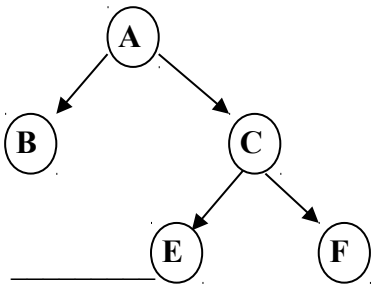
5. Suppose you implemented a queue as a Circular Linked List. Complete the enqueue method in the following code. Assume the CLLNode class defined below.

```
public class Queue{
        CLLNode tail;
        public enqueue(int data){// complete this method



      }}
class CLLNode{
    int data;
    CLLNode next;
    public CLLNode( int num,  CLLNode nxt){
            data = num; next = nxt;
    }}
```

6. Suppose you had a generic doubly-linked list as below. Complete the DLLNode definition (by filling in the blanks) and the insertAfter(here, data) method which insets a node with the given data after the node here.

```
class DLLNode<T>{

    _____ previous;

    _____  next;

    _____ data;

    DLLNode<T>(_____ prev, _____ nxt, _____ dat){
        previous = prev;
        next = nxt;
        data = dat  }}
public class DLL<T> {
    DLLNode<T> first;
    public void  insertAfter(Node<T> here, T newData{// fill in here

    }}
```

name _____



The questions from here on cover material that is not on exam 1 in 2014

7. For the tree above
   a. What is its root? _____ Its leaves ? _____
   b. What nodes are in the subtree rooted at C? _____
   c. Assume the letter written in a node is the data at that node, and assume letters are to be ordered alphabetically. Is this tree a valid Binary Search Tree? Why or why not?

8. Finish the following recursive method. Its argument is the root node of a binary tree and it should return the sum of the data elements in all of the nodes of that tree. You may assume that all data are positive integers.

```
public class BinTreeNode{
   public BinTreeNode leftSubtree, rightSubtree;
   int data;
   public static int sumData(BinTreeNode node){
      if (max == null){

          return _____ ;
    }else{

      return _____ ;
   }}
```

9. For the following class BinSchTreeNode, finish the method recBiggest so that it returns the largest data in tree. It must work recursively and efficiently. You may assume tree in not null and that it is a valid binary search tree.

```
public class BinSchTreeNode
  BinSchTreeNodeNode leftSubtree, rightSubtree;
  int data;
  public static int recBiggest(BinSchTreeNode tree){

   if (_____________________________){

     return _____________________________
   } else {
     return  _____________________________
}}
```

10. Finish the method itBiggest below. It is just like recBiggest above but it must be iterative and not recursive.

```
public int itBiggest(BinSchTreeNode tree){

 BinSchTreeNode place = _____________

 while( _____________________________){


 }
return _____________________________;}
```

11. Suppose we do the following operations on the Binary Search Tree below. Mark the resulting changes on the tree:
insert 30
insert 49
delete 66