name ___________________

CS 112 – Data Structures

Summer, 2014

Midterm Practice Exam I - Answers

NOTE: See the topic list on Sakai > Resources > Exam Info for a complete list of topics that may be on the actual exam.

- **Do not start** until everyone has an exam and the instructor tells you to begin.
- There are 6 pages in this exam, including this one. Make sure you have them all.
- This exam is closed book – closed notes.
- You must put your cellphone, PDA, Ipod, or othe electronic devices in a backpack, etc, and leave it at the front of the room. The only exception is that you can use a watch that only has time-related functions (e.g. not a calculator watch).
- Write clearly – if we can't read or can't find your answer your, answer is wrong.
- Make clear what is your answer versus intermediate work.

name ___________________

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 $O(n^2)$

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

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

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

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? n why? because in the worst case you have to compare the target with every number in the linked list

   b. best case? 1 why? because in the best case the first element of the list is >= target

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.

   a. 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? u log s

b. Suppose Algorithm B is to go through S element by element and check to see if each one is in U. What is the big-O worst case complexity of Algorithm B? u * s

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 4, pop:result 5, pop: result 7, pop: result error

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

Enqueue 4, enqueue 5, dequeue: result: _4___, enqueue 3, dequeue: result_5__

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.

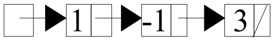
```
public class Queue{
        CLLNode tail;
        public enqueue(int data){
             if(tail ==null){
                  tail = new CLLNode(data, null);
                  tail.next = tail;
             } else {
                  tail.next = new CLLNode(data, tail.next);
                  tail = tail.next;
             }
        }}
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 inserts a node with the given data after the node here.

```
class DLLNode<T>{

    DLLNode<T>  previous;
    DLLNode<T>  next;
    T  data;
    DLLNode<T>(DLLNode<T> prev, DLLNode<T> nxt, T  dat){
        previous = prev;
        next = nxt;
        data = dat  }}
public class DLL<T> {
    DLLNode<T> first;
    public void  insertAfter(Node<T> here, T newData{// fill in below
        DLLNode<T> newNode = new DLLNode<T>(here, here.next, newData);
        if (here.next != null){
                here.next.previous = newNode; }
        here.next = newNode;
    }}
```

7. The method countPos below must be a **recursive** method that takes a Node head as its argument, goes down the list headed by head, and counts the number of nodes which have a positive data field. If the input is: head 

then the result of countPos(head) should return 2. Finish countPos.

```
public class Node{
    int data;
    Node next;
    public int countPos(Node head){

        if (head == null){
            return 0:
        } else {
            int recResult = countPos(head.next);
            if (head.data > 0){
                return recResult+1;
            } else {
                return recResult;
}}}}
```

8. In Assignment 1 you had a class of objects, ArrayLL. Two instance variables in ArrayLL, avail and numAvail, were used to keep track of which elements of array all were currently available for use in adding mere elements to the linked list represented by the ArrayLL. Method getAvail(ArrayLL a) finds the index of an available element of all, marks it as in use, and returns that index. If there is no element of all available, getAvail returns -1. Finish getAvail.

```
public class ArrayLL{
  private Item[] all;
  private int numItems;
  private int front;
  private int[] avail;
  private int numAvail;
```

```
    public int getAvail( ){
        if (numAvail == 0){
            return -1;
        } else {
            numAvail = numAvail – 1;
            return avail[numAvail];
        }
    }
}
```