SAMPLE QUESTIONS FOR MIDTERM EXAM, CS 3137

The exam is closed book, closed notes, no calculators, PDA's, cell phones, electronic devices of any kind etc.

You are responsible for any and all material covered in class, recitations, homework assignments or reading in the book up to and including Priority Queues (but not Huffman coding). The homeworks are an excellent indicator of the types of questions you will see on the exam. Below are some sample questions, but this does not cover every topic!

1. Singly Linked Lists can be implemented with the Node classes and methods below

```
public class Node {
  // Instance variables:
  private Object element;
  private Node next;
  // Simple constructors:
  public Node() {
    this(null, null);
  }
   public Node(Object e, Node n) {
    element = e;
    next = n;
  }
  // Accessor methods:
  Object getElement() {
    return element;
  }
  Node getNext() {
    return next;
  }
  // Modifier methods:
  void setElement(Object newElem) {
    element = newElem;
  }
  void setNext(Node newNext) {
    next = newNext;
  }
}
```

Assume you have a linked list of Nodes that DOES NOT use an empty header Node. Write a method

public Node reverse(Node P)

that reverses the order of the Linked List whose first node is P. The method will return a node which is the new first node of the list.

2. Write a **recursive** Java method **public Boolean sameStack(stack s1, stack s2)** to test whether two stacks contain the same elements. The elements stored in the stack are integers. The function will return true if

the stacks contain the same elements and false otherwise. No points will be given for an iterative solution. You may use the standard stack methods listed below in your method, and don't worry about how the class stack is implemented.

```
// void push( Object x )          --> Insert x
// Object pop( )             --> Remove most recently inserted item
// boolean isEmpty( )      --> Return true if empty; else false
// boolean isFull( )       --> Return true if full; else false
```

3. Prove by induction that the number of leaf nodes in a perfect binary tree of height k is 2^k.
4. Analyze the running time of binary search and show it is $O(LogN)$
5. Write a java program to solve the towers of hanoi problem. Also, give a recurrence relation which describes the algorithm's cost, and solve the recurrence to compute the time complexity.
6. Given the postfix expression $6\ 3\ +\ 8\ 4\ -\ *$, describe in **English** a stack algorithm that can be used to evalute this expression.
7. Given the infix expression: $(5 * (4 + 3 \wedge 2) * 4 - 6)$ a)Draw an expression tree for the expression, b) Write the prefix equivalent of the expression and c) Write the postfix equivalent of the expression
8. Outline an algorithm to evaluate an expression tree (pseudocode) and return the expression's numeric value.
9. Write a Java Method to: Given a list of unique integers is stored in a Doubly Linked List (in no particular order). Given a pointer to the first node in the list, delete the node containing the integer x, and return a pointer to the first node.

   public DLNode **deleteNode(DLNode first, int x)**

   . Remember that a doubly linked list Node has this format:

```
class DLNode {
  private int element;
  private DLNode next, prev;
  DLNode() { this(null, null, null); }
  DLNode(int e, DLNode p, DLNode n) {
    element = e;
    next = n;
    prev = p;
  }
  void setElement(int newElem) { element = newElem; }
  void setNext(DLNode newNext) { next = newNext; }
  void setPrev(DLNode newPrev) { prev = newPrev; }
  int getElement() { return element; }
  DLNode getNext() { return next; }
  DLNode getPrev() { return prev; }
}
```

10. Write a recursive Java method **ChildSwap(treeNode t)** that will take a pointer to a binary tree t and swap the left and right children of every node of the tree.

11. Given a pointer t to the root of a binary tree, fill in the height of each node's left and right subtrees. Store the height in each node as shown below. Note: the height of a subtree that is null can be represented as -1.

```
class Treenode{
      .
  Object data;
  Treenode left,right;
  int lheight,rheight;
      .
}
```

12. Show the trees that result from the following sequence of insertions into an AVL search tree: 100 700 400 50 20 800 900 750 500 550. You may assume you have templates for the rotations needed at your disposal (i.e. Figs. 4.31, 4.33, 4.35 and 4.36)

13. Given the following orders of traversal of a binary tree, recreate the tree:

    Postorder: G,D,B,H,I,E,F,C,A Inorder: D,G,B,A,H,E,I,C,F

14. Write a recursive Java method to test whether an binary tree satisifes the AVL balance condition. The method should return true if the tree is AVL balanced and false otherwise. Assume a standard treenode with left and right child pointers. Make sure you include code for any helper methods you use.

    public boolean isAVLBalanced(treenode root)

15. Write a Java method to number the nodes of a binary tree in level-order. Level order numbering numbers the nodes from the root to the leaves by increasing depth in the tree, left to right.

16. Given a typical Heap:

```
public MyBinaryHeap( int capacity )
{
     currentSize = 0;
     array = new Comparable[ capacity + 1 ];
}
```

Write an insert(Object X) method for the heap.

17. Below is a short listing of some student and class data for fall registration.

Name	Courses			
Smith	CS 3137	CS 4107	CS 3251	CS 3823
Jones	CS 3137	CS 4111	CS 3062	CS 3823
Pellet	CS 1007	CS 4107	CS 3823	
Farmer	CS 1007	CS 4111	CS 4241	CS 3823
Powell	CS 3062	CS 4241	CS 3137	CS 1007
Kelly	CS 3107	CS 4107	CS 3251	CS 3823
Chen	CS 3137	CS 4111	CS 3062	CS 3823
Maran	CS 1007	CS 3107	CS 3823	
Boos	CS 3137	CS 4107	CS 4241	CS 3823
Larmer	CS 3062	CS 4241	CS 3251	CS 1007

Describe (don't program!) the data structures you would create so that an administrator could print out a list of each student and his/her classes, and a list of students in each class.

Using the data structures you implemented above, describe how a Drop/Add program would work. Input to the drop/add would be as below:

Name	Drop/Add	Course
Kelly	Drop	CS 3823
Kelly	Add	CS 4111
Maran	Drop	CS 3823
Powell	Add	CS 3824
Maran	Drop	CS 3107
Maran	Drop	CS 1007