CSA 274-A FINAL EXAM 2006

Miami University - CSA274-A - Spring 2006 - Final Exam
There are 15 questions for a total of 210 points.

Name: ____

Instructions

Please read through this entire exam very carefully before starting.

You may use one sheet of paper no larger than size 8.5" by 11" in size, double sided, of pre-prepared material. This sheet must be prepared by you and not shared with anyone else.
Other than that, this exam is closed notes and closed books.

All work must be written on the exam pages in order to be graded. Any scrap paper used, must be the scrap paper provided during the exam period.

For programming questions: Please be as accurate as possible with your Java syntax: this includes appropriate use of braces, semicolons, and the proper use of upper/lowercase letters.

No electronic devices may be used during the exam: this includes calculators, iPods, PDAs, and cellular phones.

You have 120 minutes to complete the exam.

There are 210 possible points, but the exam will be graded out of 200 points.

Good Luck!

1. Given the following code for generic data structure:

```
public class Building<E,T> {
    private E entrance;
    private T elevator;

    public Building() {}

    // .. remaining valid class definition
}
```

   (a) (3 points) The symbols `E` and `T` represent what? ________________

   (b) (3 points) Write the line of code to declare a variable named `kregerHall` of type `Building` where the generic class types used are `DoubleDoor` and `Slow`, the variable should be initialized by invoking the default constructor.

2. (15 points) For array list, we discussed dynamically growing the list in the event that more space is needed. It might also make sense to shrink the array to reclaim unused space. Complete the `shrink` method below to reclaim space *in the event that the items in the array are less than half of the capacity*, by reducing the capacity by half of the current capacity.

```
public class ArrayList<E> {
    private E[] data;
    private int size;

    // assume the rest of the class is listed and complete

    public E removeLast() {
        E rtn = data[--size];
        shrink();
        return rtn;
    }

    private void shrink() {
        // BEGIN ANSWER









        // END ANSWER
    }
}
```

3. (20 points) Tree traversals apply not only to binary trees and/or search trees, but to all trees. For example, consider the tree in Figure 1. This is a tree in which any given node can have a variable number of children. Traversals can also be performed using a stack data structure. For example, to perform a **preorder** traversal on a tree we can `pop` the top node off of the stack, process it, `push` all of the node's children on the stack (in reverse order) and repeat the process. To bootstrap the algorithm, we push the root on to the stack and start the process. A preorder traversal using a stack is an **iterative** process and is *not recursive.*

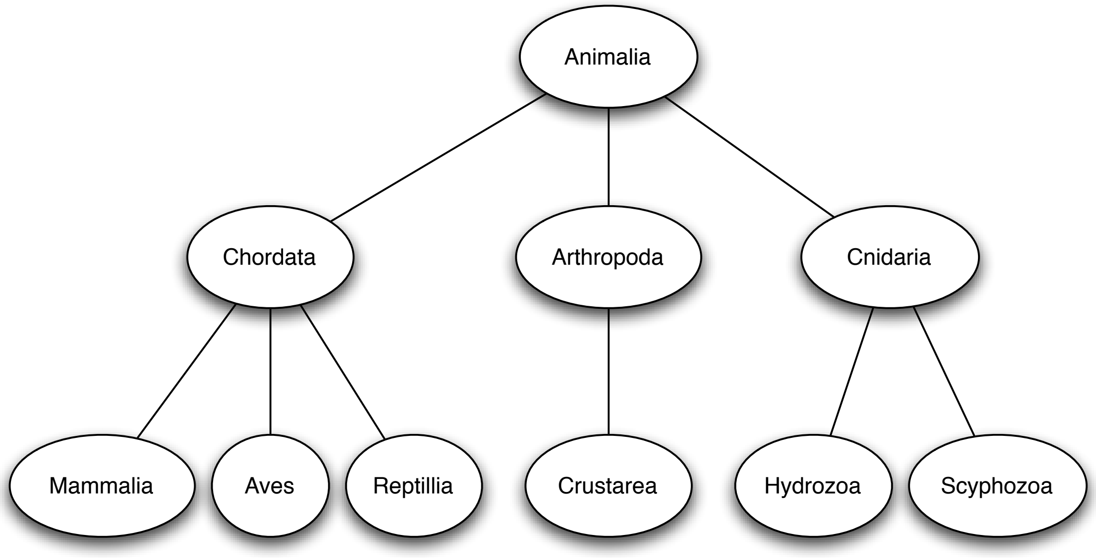


Figure 1: Animal Kingdom Tree

For this question, you are to complete the iterative preorder traversal method using only what is provided (next page).

```
public class GenericTree<E> {
    private Node<E> root;

    private static class Node<E> {
        private E data;
        private List<Node<E>> children = (List<Node<E>>) new ArrayList();
    }

    public List<E> preOrderTraversal() {
        List<E> results = new ArrayList<E>();

        // Stack has methods push(E), pop(), and isEmpty() - ignore exceptions
        Stack<Node<E>> stack = new Stack<Node<E>>();
        stack.push( root );
        // BEGIN ANSWER




























        // END ANSWER
        return results;
    }
}
```

4. (15 points) Code a recursive method for converting a **string** of digits into the integer it represents. For example "1326458" represents the integer 1,326,458. For this; the input is a `String`, the output is an `int`, and your method should be named `convertInt`.

```
public class Converter {

private int convertDigit( char c ) {
// ASSUME THAT THIS METHOD WORKS FOR CONVERTING A SINGLE DIGIT
}

// BEGIN ANSWER - you need to define the method(s) used











    // END ANSWER
}
```

5. Rank the following growth rates from the slowest rate of growth first to the fastest rate of growth (such that $O(a) < (b) < (c)$ etc...): $O(\log n), O(n), O(1), O(n^2), O(n!), O(n \log n)$

   (a) (2 points) ____

   (b) (2 points) ____

   (c) (2 points) ____

   (d) (2 points) ____

   (e) (2 points) ____

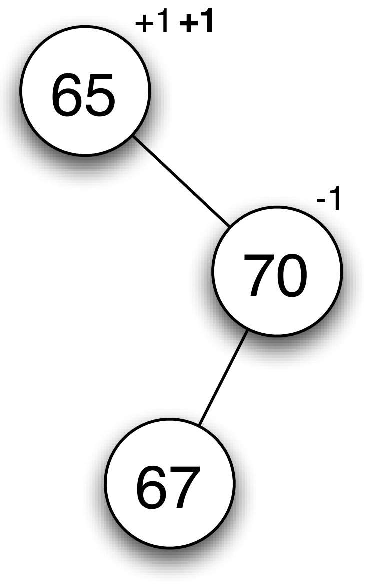
   (f) (2 points) ____

6. (5 points) Convert the following code from explicit iterator use to Java (enhanced for loop.

```
List<Integer> list = new ArrayList<Integer>();
fillList( list );

// START converting here
Iterator<Integer> iter = list.iterator();
while( iter.hasNext() ) {
   Integer thisInt = iter.next();
         System.out.println( thisInt );
}
// END converting here
```

7. (5 points) For a hash table of size n, using open addressing, and assuming that resizing is disabled. What is the worst case running time for inserting into the hash table? Why? *You may assume that the table will never allow more than $n-1$ elements in it.*

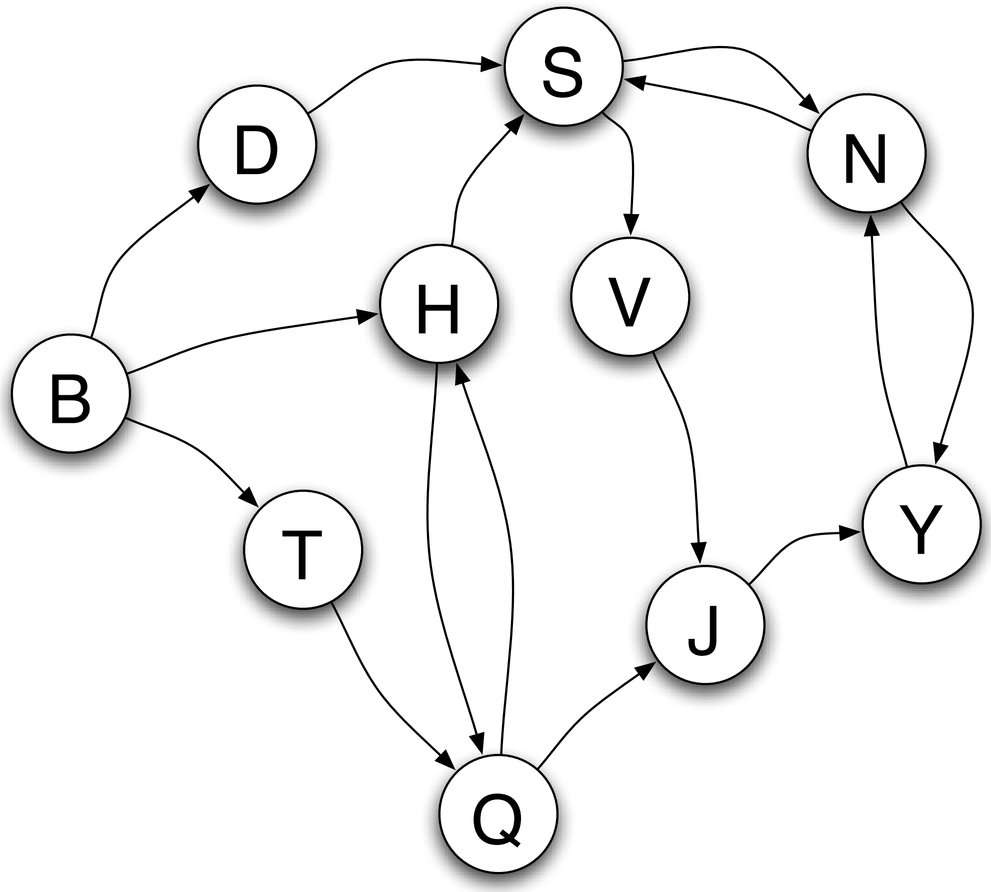
8. (5 points) The AVL tree shown below has just had 67 inserted. Show the proper rotation(s) to return the tree to balance, all nodes should have a balance of 0 with the process is complete.



9. (10 points) Using insertion sort, show the status of the array after each pass through the algorithm.

53	33	21	2	72	14	14	40

10. (12 points) Show the depth first search tree starting at node B. Vertices should be selected in alphabetical order when there is more than 1 to choose from. *Vertices should be processed in the order in which they are discovered.*

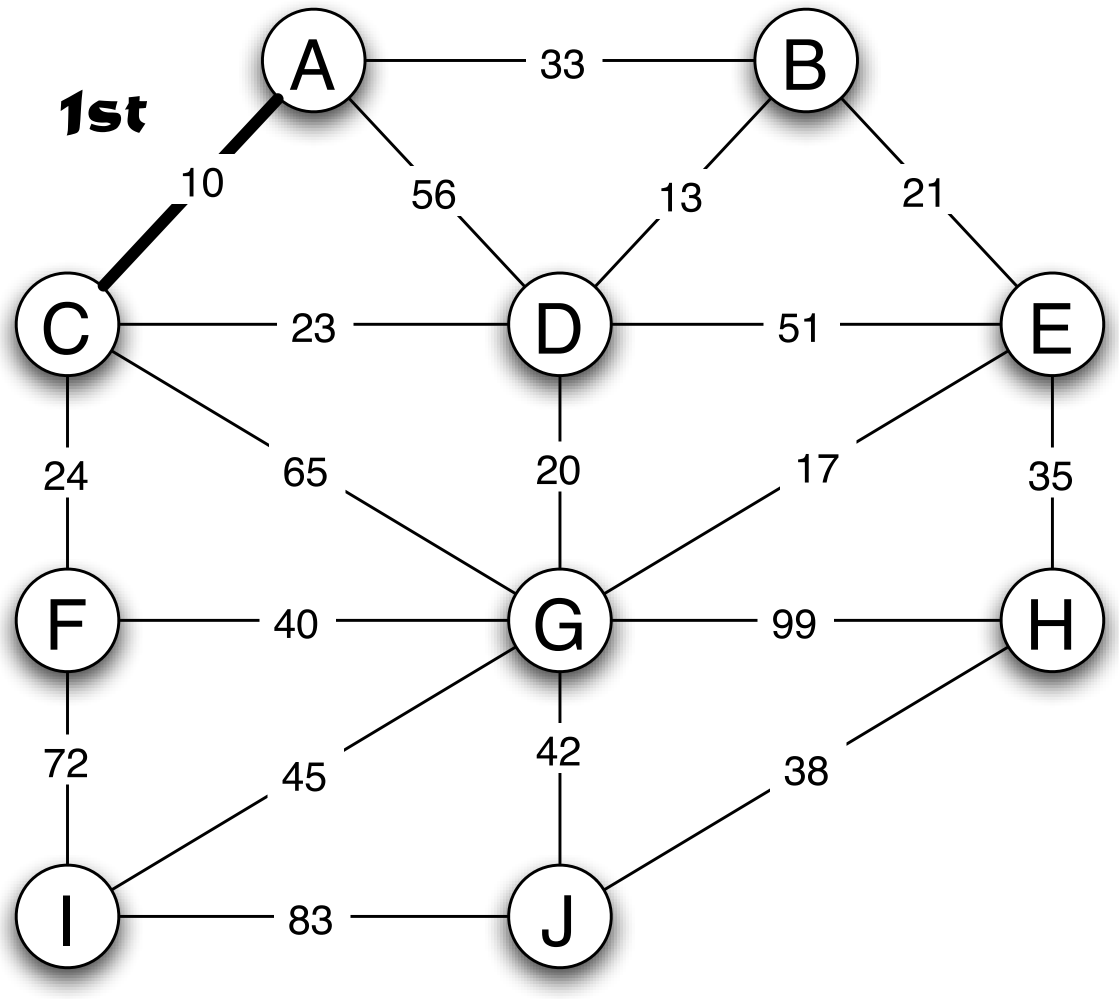


11. (20 points) We have discussed Prim's algorithm for building a minimum spanning tree. With Prim's algorithm we grow a tree by selecting the next vertex that is closest to the current tree.

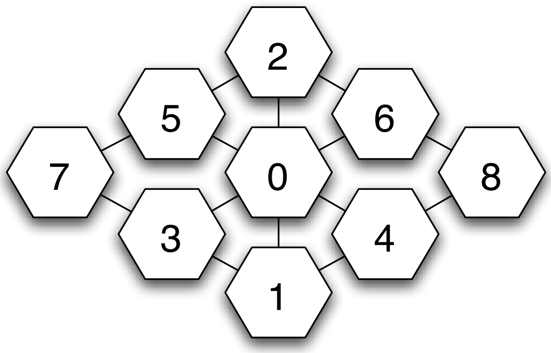
    Kruskal's algorithm is another important algorithm for calculating the minimum spanning tree in a graph. Kruskal's algorithm works by selecting the *edge* with the lowest weight that (1) is not in the tree and (2) will not cause a cycle. For example, in the graph below, the edge from $A \rightarrow C$ will be the first taken because it has the lowest weight of the unselected edges (initially all edges) and does not create a cycle.

    A cycle is created when the edge to be added would be between two vertices that are already in the minimum spanning tree. For Kruskal's algorithm, if an edge were to create a cycle, we simply disregard it and move on to the edge with the next lowest weight.

    For this question, show the minimum spanning tree in the below graph using Kruskal's algorithm. Answer by indicating the order in which the edges are chosen to join the minimum spanning tree.



12. You are working on a new computer game with a team of programmers. While the game is nearly complete, there is a problem with the game play. The board is a undirected graph taking the form shown below. A player can move their playing piece along any of the edges during their turn. One of the programmers on your team informs you that while playing the



game, a player only seems to be able to move from one space to another if the number on the **from square** is *lower* than the number on the **to square**. After running your JUnit tests on the game board class, you see that they are no longer working, so you print out the adjacency matrix that represents the graph of the board, and see this (and empty space indicates 0 or no edge, and 1 represents the presence of an edge):

X	0	1	2	3	4	5	6	7	8
0		1	1	1	1	1	1		
1				1	1				
2						1	1		
3								1	
4									1
5								1	
6									1
7									
8									

(a) (5 points) What's wrong with the graph representation in memory?

(b) (25 points) The team agrees on a quick fix where you will take the game board and correct the representation in memory using a small utility function (which they also ask you to write). Fill in the details of this method on the following page.

```
public class GameBoard {
  private static final int SIZE = 9;
  private int[][] board = new int[size][size];

  public GameBoard() {
    initializeBoard();
    correctBoard(); // this is the method that you need to write
  }

  // ... the rest of the class

  private void correctBoard() {
    // BEGIN ANSWER

    // END ANSWER
  }
}
```

13. Merge sort can be easily split into two methods, a recursive merge sort method and a utility method to merge two sub-arrays back into a source array. Complete the code for merge sort where the input array consists only of items that implement the `Comparable` interfaces (that is, you must use the `compareTo` method to determine their order).

    (a) (10 points) Complete the code for the sort method.

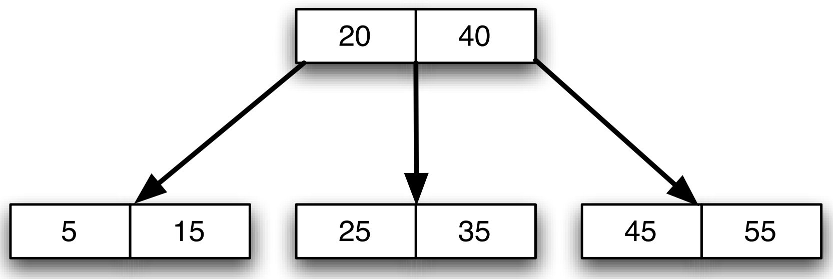
    (b) (15 points) Complete the code for the merge method.

```
public class MergeSort {
  public static  <T extends Comparable<T>> void sort( T[] array ) {










  }
  private static <T extends Comparable<T>> void merge(T[] output, T[] left, T[] right) {
















  }
}
```

14. (15 points) Write a recursive method for finding the largest item in this linked list, noting that all items are *comparable* and thus have a *compareTo* method.

```
public class LinkedList<E extends Comparable> {
    private Node<E> root;

    private static class Node<E2 extends E> {
        private E2 data;
        private Node<E2> next;
    }

    public E findLargest() {
        return findLargest(root);
    }

    private E findLargest( Node<E> root ) {
        // BEGIN ANSWER




















        // END ANSWER
    }
}
```

15. (15 points) Given the 2-3 Tree below, show the resultant tree if the value `30` is inserted. You may either show the final tree or any intermediate steps that you find helpful.


20
40
5
15
25
35
45
55