Name:_____

15-111/Kesden Spring 2003
Exam 2 (Retake)

Basic Java

1. Please consider the following `Person` class. Notice the use of the Java keyword "this". What is "this"? And why is it used in the constructor below?

```
class Person {
private String fullName;
private String fullAddress;

public Person (String fullName, fullAddress) {
        this.fullName = fullName;
        this.fullAddress = fullAddress;
}

public String getFullname() {
        return fullName;
}

public String getFullAddress() {
        return fullAddress;
}
}
```

2. Below is the skeleton of a `PersonFinder` class based on the `Person` class defined above. Please complete this skeleton using a `Vector` as the primary data structure.

```
class PersonFinder {

// Your code here

// Constructor
public PersonFinder () {
        // Your code here
}

// Return a Person's fullAddress
public String findAddress(String fullName) {
        // Your code here
}


// Add a Person to this PersonFinder
public void addPerson (Person newEntry)
{
        // Your code here
}
}
```

Singly Linked Lists

3. Given the provided, minimal *LinkedList*, including the *Node*, please implement the method described below:

```
/**
  Removes the first item equal to the keyItem from the list.
  This method removes at most one item. If there are multiple
  matching items, it removes only the matching item closest to the
  head. The list is not changed in any other way.
  <p>
  In the event of an error, it does not change the list, instead it
  returns leaving the list in its prior condition.
  <p>
  @param  keyItem  The first item within the list equal to this item
  is removed from the list.
 */
 public void removeFirstMatchingItem (Comparable keyItem) {

   return;
 }
```

4. Given the provided, minimal *LinkedList* class, including the *Node*, please implement the method described below:

```
/**
  Creates and returns a new list, which contains exactly those
  items that are present in exactly one of the two lists, but
  not both.
  <p>
  It does not change either of the original lists
  <p>
  In the event of an error, it returns an empty list.
  <p>
  @param  otherList is the list that should be compared with
  this list
  <p>
  @return  a new list, which contains exactly those items
  that are present in exactly one of the two lists, but not
  both.
 */
 public LinkedList Xor(EnhancedLinkedList otherList) {

   return null; // Remove this!
 }
```

Queues

5. Please draw a figure, or a collection of figures, that shows the evaluation of the following expression using a single stack. The figure(s) should depict the stack after each operation and should also clearly indicate the operation.

   4,5,+,3,/,2,*

```
public class LinkedList
{

        public class IndexException extends Exception
        {

                public String toString()
                {
                        return ("Bad index in Linked List");
                }
        }


        private Node head;
        private Node tail;



        public LinkedList()
        {
                head = tail = null;
        }


        public void addHead(Comparable data)
        {

                Node newNode;
                newNode = new Node(data);

                newNode.setNext(head);

                head = newNode;

                if (null == tail)
                {
                        tail = head = index = newNode;
                }
        }



}
```

```
// For question #1 and question #2
public class Node
{
        private Comparable data;
        private Node next;



        public Node (Comparable data, Node next)
        {
                this.data = data;
                this.next = next;
        }


        public Node (Comparable data)
        {
                this.data = data;
                this.next = null;
        }

        public Comparable getData()
        {
                return data;
        }


        public Node getNext()
        {
                return next;
        }



        public void setNext(Node next)
        {
                this.next = next;
        }



        public void setData(Comparable data)
        {
                this.data = data;
        }
}
```