CMP-338 110 Points Total Final Fall 2015

Version 1

Instructions

- **Write your name and version number on the top of the yellow paper.**
- **Answer all questions on the yellow paper.**
- **One question per page.**
- **Use only one side of the yellow paper.**

1. (20 Points) Multiple Choice:

A. (2 Points) In a graph, all ______ begin and end at the same vertex and do not pass through any other vertices more than once.
   a. Paths
   b. Simple paths
   c. Cycles
   d. Simple cycles

B. (2 Points) A connected undirected graph that has n vertices must have at least ______ edges.
   a. n
   b. n – 1
   c. n / 2
   d. n * 2

C. (2 Points) The maximum height of a binary search tree of n nodes is ______.
   a. n
   b. n – 1
   c. n / 2
   d. $\log_2(n + 1)$

D. (2 Points) In an array based representation of a complete binary tree, which of the following represents the left child of node tree[i]?
   a. tree[i+2]
   b. tree[i–2]
   c. tree[2*i+1]
   d. tree[2*i+2]

E. (2 Points) In an array based representation of a complete binary tree, which of the following represents the parent of node tree[i]?
   a. tree[i–2]
   b. tree[(i–1)/2]
   c. tree[2*i–1]
   d. tree[2*i–2]

F. (2 Points) The quicksort is _____ in the worst case.
   a. $O(n^2)$
   b. $O(n^3)$
   c. $O(n * \log_2 n)$
   d. $O(\log_2 n)$

G. (2 Points) A method in a subclass is said to ______ an inherited method if it has the same method declarations as the inherited method.
   a. Copy
   b. Override
   c. Overload
   d. Cancel

H. (2 Points) The ______ access modifier hides the members of a class from the class's clients but makes them available to a subclass and to another class within the same package.
   a. public
   b. private
   c. protected
   d. package access

I. (2 Points) Which of the following code fragments is used to delete the item at the front of a queue represented by a circular array?

```
a. front=MAX_QUEUE - front;
       --count;
b. front=front - back;
       --count;
c. front=(front+1)%MAX_QUEUE;
       --count;
d. front=(back+1)% MAX_QUEUE;
       --count;
```

J. (2 Points) If the array: {6, 2, 7, 13, 5, 4}is added to a stack, in the order given, which number will be the first number to be removed from the stack?
   a. 6
   b. 2
   c. 5
   d. 4

2. (20 Points) Re-write the following QuickSort Class and fix all 10 **logical** errors:

```
import java.util.Vector;
public class QuickSort {
    public static <T extends Comparable<? super T>> void quickSort(Vector<T> theVector,
                                                                    int first, int last) {
        if (first < last) {
            int pivotIndex = partition(theVector, first, last);
            quickSort(theVector, first, pivotIndex + 1);
            quickSort(theVector, pivotIndex - 1, last);
        } // end if
    } // end quicksort

    public static <T extends Comparable<? super T>> void choosePivot(Vector<T> theVector,
                                                                      int first, int last) {

        // The pivot will be the middle value of first, mid and last
        int mid = (first + last) / 2;
        T temp = theVector.elementAt(first);
        T f = theVector.elementAt(first);
        T m = theVector.elementAt(mid);
        T l = theVector.elementAt(last);

        if (((f.compareTo(m) <= 0) || (l.compareTo(m) >= 0)) &&
            ((f.compareTo(m) >= 0) || (l.compareTo(m) <= 0))) {
            theVector.set(first, theVector.elementAt(mid));
            theVector.set(mid, temp);
        } else if (((f.compareTo(l) <= 0) || (m.compareTo(l) >= 0)) &&
                  ((f.compareTo(l) >= 0) || (m.compareTo(l) <= 0))) {
            theVector.set(first, theVector.elementAt(last));
            theVector.set(last, temp);
        }
    } // end choosePivot

    public static <T extends Comparable<? super T>> int partition(Vector<T> theVector,
                                                                int first, int last) {
        T tempItem;
        choosePivot(theVector, first, last);
        T pivot = theVector.elementAt(first); // reference pivot
        int lastS1 = first; // index of last item in S1
        for (int firstUnknown = first + 1; firstUnknown <= last; --firstUnknown) {
            if (theVector.elementAt(firstUnknown).compareTo(pivot) < 0) {
                --lastS1;
                tempItem = theVector.elementAt(firstUnknown);
                theVector.set(firstUnknown, theVector.elementAt(lastS1));
                theVector.set(lastS1, tempItem);
            } // end if
        } // end for
        tempItem = theVector.elementAt(first);
        theVector.set(first, theVector.elementAt(lastS1));
        theVector.set(lastS1, tempItem);
        return lastS1;
    } // end partition
}
```

3. (50 Points) Given the following BinarySearchTreeInterface, TreeItem, and TreeNode implementations. Write the complete Java class for the BinarySearchTree which implements the given BinarySearchTreeInterface.

```java
import java.util.Vector;
public interface BinarySearchTreeInterface {
    // returns true if BinarySearchTree is empty,
    // false otherwise
     public boolean isEmpty();

    // makes the BinarySearch Tree empty
     public void makeEmpty();

    // inserts the given item into the
    // BinarySearchTree
     public void insert(TreeItem item);

    // retrieves the TreeItem with the given key
     public TreeItem retrieve(int key);

    // returns a Vector of TreeItems using inorder
    // traversal
     public Vector<TreeItem> inorder();

    // compares the given BinarySearchTree to this
    // one and returns true if they have equal
    // TreeItems
     public boolean equals(Object o);
}

public class TreeItem implements Comparable {
     private int item;

     public TreeItem(int item) {
          this.item = item;
    }

    public int compareTo(Object o) {
        TreeItem ti = null;
        if (o instanceof TreeItem) {
            ti = (TreeItem)o;
        }
        if (this.item == ti.getItem()) {
            return 0;
        } else if (this.item > ti.getItem()) {
            return 1;
        } else {
            return -1;
        }
    }

    public int getItem() {
        return item;
    }

    public boolean equals(Object o) {
        if (this.compareTo(o) == 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

```java
public class TreeNode {
    private TreeItem item;
    private TreeNode leftChild = null;
    private TreeNode rightChild = null;

    public TreeNode(TreeItem item) {
        this.item = item;
    }

    public TreeItem getItem() {
         return item;
    }

    public TreeNode getLeftChild() {
         return leftChild;
    }

    public void setLeftChild(TreeNode leftChild) {
        this.leftChild = leftChild;
    }

     public TreeNode getRightChild() {
        return rightChild;
    }

    public void setRightChild(TreeNode rightChild) {
        this.rightChild = rightChild;
    }
}
```

4. (20 Points) Given the following list of numbers: 40, 30, 55, 20, 10, 50, 70, 65, 5, 15, 4, 60, 85, 54, 8 being inserted in the given order.
    a. (5 Points) Draw the resulting Binary Search Tree.
    b. (5 Points) Draw the resulting 2-3 Tree.
    c. (5 Points) Draw the resulting 2-3-4 Tree.
    d. (5 Points) What order should the numbers be inserted in order to obtain a Full Binary Search Tree?

CMP-338 | 110 Points Total | Final Fall 2015

Version 2

Instructions

- **Write your name and version number on the top of the yellow paper.**
- **Answer all questions on the yellow paper.**
- **One question per page.**
- **Use only one side of the yellow paper.**

1. (20 Points) Multiple Choice:

A. (2 Points) Which of the following is true about a simple cycle in a graph?
   a. It can pass through a vertex more than once
   b. It can not pass through a vertex more than once
   c. It begins at one vertex and ends at another vertex
   d. It passes through only one vertex

B. (2 Points) A tree with n nodes must contain _______ edges.
   a. n
   b. n – 1
   c. n / 2
   d. n * 2

C. (2 Points) Locating a particular item in a binary search tree of n nodes requires at most _______ comparisons.
   a. n
   b. n * 3
   c. n / 2
   d. n – (n / 2)

D. (2 Points) In an array based representation of a complete binary tree, which of the following represents the right child of node tree[i]?
   a. tree[i+2]
   b. tree[i–2]
   c. tree[2*i+1]
   d. tree[2*i+2]

E. (2 Points) In an array based representation of a complete binary tree, which of the following represents the parent of node tree[i]?
   a. tree[i–2]
   b. tree[(i–1)/2]
   c. tree[2*i–1]
   d. tree[2*i–2]

F. (2 Points) The quicksort is _____ in the worst case.
   e. $O(n^2)$
   f. $O(n^3)$
   g. $O(n * \log_2 n)$
   h. $O(\log_2 n)$

G. (2 Points) The keyword _______ is used in the class declaration of a subclass to indicate its superclass.
   a. inherits
   b. extends
   c. implements
   d. super

H. (2 Points) A class's _____ members can only be used by its own methods.
   a. public
   b. protected
   c. private
   d. package access

I. (2 Points) Which of the following is the code to insert a new node, referenced by `newNode`, into an empty queue represented by a circular linked list?

```
a. newNode.setNext(lastNode);
b. lastNode.setNext(lastNode);
     lastNode = newNode;
c. newNode.setNext(lastNode);
     newNode = lastNode;
d. newNode.setNext(newNode);
     lastNode = newNode;
```

J. (2 Points) If the array: {6, 21, 35, 3, 6, 2, 13} is added to a stack, in the order given, which of the following is the top of the stack?
   a. 2
   b. 6
   c. 35
   d. 13

2. (20 Points) Re-write the following MergeSort Class and fix all 10 **logical** errors:

```
import java.util.Vector;
public class MergeSort {
    public static <T extends Comparable<? super T>> void sort(Vector<T> theVector ) {
        Vector<T> tempVector = new Vector<T>(theVector.size());
        for ( int i = 0 ; i <= theVector.size() ; i++ ) {
            tempVector.add(null);
        }
        mergeSort(theVector, tempVector, 0, (theVector.size() + 1));
    }

    public static <T extends Comparable<? super T>> void mergeSort(Vector<T> tempVector,
                                                                       Vector<T> theVector,
                                                                       int first, int last) {

        if (first < last) {
            int mid = (first + last) / 2; // index of midpoint
            mergeSort(theVector, tempVector, first, mid + 1);
            mergeSort(theVector, tempVector, mid - 1, last);
            merge(theVector, tempVector, first, mid, last);
        } // end if
    }

    public static <T extends Comparable<? super T>> void merge(Vector<T> theVector,
                                                                   Vector<T> tempVector,
                                                                   int first, int mid, int last) {

        int first1 = first;
        int last1 = mid;
        int first2 = mid + 1;
        int last2 = last;
        int index = first1;
        while ((first1 <= last1) && (first2 <= last2)) {
            if (theVector.elementAt(first1).compareTo(theVector.elementAt(first2)) > 0) {
                tempVector.set(index, theVector.elementAt(first2));
                first1++;
            } else {
                tempVector.set(index, theVector.elementAt(first1));
                first2++;
            } // end if
            index++;
        } // end while
        while (first1 <= last1) {
            tempVector.set(index, theVector.elementAt(first1));
            first1++;
            index++;
        } // end while
        while (first2 <= last2) {
            tempVector.set(index, theVector.elementAt(first2));
            first2++;
            index++;
        } // end while

        for (index = first; index < last; ++index) {
            theVector.set(index, tempVector.elementAt(index));
        } // end for
    } // end merge
}
```

3. (50 Points) Given the following BinarySearchTreeInterface, TreeItem, and TreeNode implementations. Write the complete Java class for the BinarySearchTree which implements the given BinarySearchTreeInterface.

```
import java.util.Vector;
public interface BinarySearchTreeInterface {
    // returns true if BinarySearchTree is empty,
    // false otherwise
     public boolean isEmpty();

    // makes the BinarySearch Tree empty
     public void makeEmpty();

    // inserts the given item into the
    // BinarySearchTree
     public void insert(TreeItem item);

    // retrieves the TreeItem with the given key
     public TreeItem retrieve(int key);

    // returns a Vector of TreeItems using inorder
    // traversal
     public Vector<TreeItem> inorder();

    // compares the given BinarySearchTree to this
    // one and returns true if they have equal
   // TreeItems
     public boolean equals(Object o);
}

public class TreeItem implements Comparable {
     private int item;

     public TreeItem(int item) {
          this.item = item;
    }

    public int compareTo(Object o) {
        TreeItem ti = null;
        if (o instanceof TreeItem) {
            ti = (TreeItem)o;
        }
        if (this.item == ti.getItem()) {
            return 0;
        } else if (this.item > ti.getItem()) {
            return 1;
        } else {
            return -1;
        }
    }

    public int getItem() {
        return item;
    }

    public boolean equals(Object o) {
        if (this.compareTo(o) == 0) {
            return true;
        } else {
            return false;
        }
    }
}
```

```
public class TreeNode {
    private TreeItem item;
    private TreeNode leftChild = null;
    private TreeNode rightChild = null;

    public TreeNode(TreeItem item) {
        this.item = item;
    }

    public TreeItem getItem() {
         return item;
    }

    public TreeNode getLeftChild() {
         return leftChild;
    }

    public void setLeftChild(TreeNode leftChild) {
        this.leftChild = leftChild;
    }

     public TreeNode getRightChild() {
        return rightChild;
    }

    public void setRightChild(TreeNode rightChild) {
        this.rightChild = rightChild;
    }
}
```

4. (16 Points) Given the following list of numbers: 50, 45, 30, 65, 60, 35, 20, 55, 63, 10, 5, 68, 70, 80, 15 being inserted in the given order.
   a. (4 Points) Draw the resulting Binary Search Tree.
   b. (4 Points) Draw the resulting 2-3 Tree.
   c. (4 Points) Draw the resulting 2-3-4 Tree.
   d. (4 Points) What order should the numbers be inserted in order to obtain a Full Binary Search Tree?