CMPSCI 187

Final Exam

December 18, 2015

100 Points

120 Minutes

The first part of the exam should be answered using an opscan form. Make sure you leave enough time for the four programming questions.

Please enter your student id and name on the opscan form NOW.

Good luck!

Name: _____

Student ID: _____

Problem:	1	2	3	4	5	Total
Maximum:	40	15	15	15	15	100
Score:	[Opscan]					

Do not circle answers on these pages! You must enter answers on the opscan form.

I. Multiple Choice and True / False questions

1) Which of the following is **not** a required condition for the binary search algorithm if we want to achieve O(log n) for search?

a. The list must be sorted

b. There should be direct access to the middle element in any sublist

c. There must be a mechanism to delete and/or insert elements in list

d. none of above

2) If the in-order traversal of a binary tree resulted in E A C K F H D B G; the postorder traversal would be:

a. EKCAHGBDF
b. FAEKCDHGB
c. EAFKHDCBG
d. ECAFHGBDK
e. more than one is possible

3) The best case running time for insertion sort is O(n).

a. True
b. False

4) When using the Java Collections class, the following statement compiles:

List<T> foobar = new List<T>();

a. True
b. False

5) Suppose that we have an undirected graph with six vertices, such that the graph is **connected**. What is the smallest number of edges the graph could have?

a. 5
b. 8
c. 32
d. 64

6) What are the correct intermediate steps when the following data set is being sorted with the bubble sort? List the sequences you get after performing each required **swap**.

15, 20, 10, 18

a. 15, 10, 20, 18 -- 15, 10, 18, 20 -- 10, 15, 18, 20
b. 10, 20, 15, 18 -- 10, 15, 20, 18 -- 10, 15, 18, 20
c. 15, 10, 20, 18 -- 10, 15, 20, 18 -- 10, 15, 18, 20
d. 15, 18, 10, 20 -- 10, 18, 15, 20 -- 10, 15, 18, 20

7) Which has better worst case complexity: Quick Sort or Insertion Sort?

a. Insertion Sort
b. Quick Sort
c. They have the same worst case complexity

8) If a heap is used to implement a priority queue, what is the big-O efficiency of the enqueue operation, assuming the size of the priority queue is N?

a. O(n)
b. O(log n)
c. O(n^2)
d. O(2^n)
e. None of the above

9) Imagine you are using a stack to evaluate a post fix expression. So far you have evaluated this part of the expression: 9 3 - 1 4 + 7 4 2 *. Which of the following illustrates the state of the stack at this point in the evaluation?

a) bottom: -6 1 4 11 2
b) bottom: -6 5 7 8
c) bottom: 6 5 7 8
d) bottom: 6 5 11 4 2
e) None of the above.

10) You can implement the binary search algorithm on an array that is sorted in reverse order.

a) True
b) False

11) If you perform Breadth First Search on a tree you don't need to track the nodes you've visited.

a)True
b) False

12) What is the complexity of the following method?

```
public static int fib(int n){

      if(n == 0) return 0;

      if(n == 1) return 1;

      return fib(n - 1) + fib(n - 2);

}
```

a. O(n)
b. O(log n)
c. O(n^2)
d. O(2^n)
e. None of the above

13) What exception gets thrown from this code (assume all exceptions are unchecked)?

```
try {
      throw new TryException();
}
catch (TryException e) {
      throw new CatchException();
}
```

a) TryException
b) CatchException
c) none of those exceptions
d) doesn't compile

14) What does this code do?

```
public class GenericItem <E> {
      E item;
      public GenericItem(E s) {
            item = s;
      }
      public Integer is10(){
            String s = "10";
            return item.compareTo(s);
      }
}
GenericItem<String> gc = new GenericItem<String>("10");
System.out.println(gc.is10());
```

a) prints true
b) Throws NullPointerException
c) Throws GenericItem exception
d) Does not compile
e) Throws Parsing exception

15) Given a binary search tree T, which of the following could result from the in-order traversal of T (assume the binary search tree uses lexicographic (alphabetical) ordering)?

a) GADHKCT
b) TKHGDCA
c) ACDGHKT
d) GACDHKT
e) Any of the above

For the following two questions:
Imagine you have two lists of items, L1 and L2, and a single hash function H which hashes items into one of four buckets. The items in each bucket of the hash tables are shown as dots below:

	Bucket 1	Bucket 2	Bucket 3	Bucket 4
L1 hash table:	●●●	[empty]	●	●
L2 hash table	●	●●	●	●●●

So, L1 has a total of 5 items and L2 has a total of 7 items. And, for example, you know that 3 items from L1 hash to Bucket 1.

Given the contents of the hash tables above, and the fact that both lists are hashed with the **same** hash function. Answer the following two questions:

16) What is the **minimum** number of items that could be in the intersection of L1 and L2 (that is, what is the smallest number of items L1 and L2 could have in common)?

a. 0
b. 2
c. 3
d. 5
e. 7

17) What is the **maximum** number of items that could be in the intersection of L1 and L2 (that is, what is the largest number of items that L1 and L2 could have in common)?

a. 2
b. 3
c. 5
d. 7
e. 12

18) Which of the following could replace < Comparison method header > to make this class compile with no errors?

```
public class PriorityItem implements Comparable
{

    < Comparison method header > { < Code not shown > }

}
```

I. public int compare(PriorityItem item1, PriorityItem item2)

II. public boolean compareTo(PriorityItem item1)

III. public int compareTo(Object obj)

a. I only
b. II only
c. III only
d. I or II
e. II or III

19) A heap element that is not the root and that is at position INDEX in the array has a parent at what position in the array?

a) (INDEX−1) / 2
b) (INDEX*2) + 1
c) (INDEX*2) + 2
d) (INDEX*2) + 3
e) (INDEX) / 2

20) What is the output of executing the main() method below?

```
class Nested {

      public static void checkInteger(int i) throws Exception {
            System.out.print("U");
            if (i > 0)
                  throw new Exception("V");
            System.out.print("W");
      }

      public static void main(String[] args) {
            try {
                  System.out.print("X");
                  checkInteger(100);
                  System.out.print("Y");
            }
            catch(Exception e) {
                  System.out.print( e.getMessage() );
            }
            System.out.print("Z");
      }
}
```

a) XUVW
b) XUVWYZ
c) XUVZ
d) XUWYVZ
e) XUYVZ

PROGRAMMING QUESTIONS

For problems I and II, write **recursive** methods to check two properties of a binary tree. **You may only use the methods and instance variables for the classes defined below**. You may write helper methods.

```
public class TNode<T extends Comparable<T>> {
	private T data; private TNode<T> left; private TNode<T> right;
	public TNode(T data, TNode<T> left, TNode<T> right) {
		this.data = data;
		this.left = left;
		this.right = right;
	}
	public T getData() { return data; }
	public void setData(T data) { this.data = data; }
	public TNode<T> getLeft() { return left; }
	public void setLeft(TNode<T> left) { this.left = left; }
	public TNode<T> getRight() { return right; }
	public void setRight(TNode<T> right) { this.right = right; }
}


public class Tree<T extends Comparable<T>> {

	protected TNode<T> root;

	public int size (){ // You may assume this is implemented correctly}

	boolean isHeapOrdered(){ PART I }
	boolean isComplete(){ PART II }
}
```

I) Write a **recursive** method `isHeapOrdered` to check if the values of the tree conform to the max-heap ordering rule.

```
boolean isHeapOrdered(){
```

II) Write a **recursive** method `isComplete` to check that the tree is complete. Hint: Think about the size of the tree and how to compute the index of a node recursively using a helper method.

```
boolean isComplete(){
```

III) Given the following GraphInterface, write an **iterative** method hasCycleFrom to detect if the graph has any cycles that include the given vertex. Return true if there is a cycle in the graph involving the vertex v, otherwise return false. You may use the Java Collections classes.

```
public interface GraphInterface<T extends Comparable>
{
 boolean isEmpty();
 boolean isFull();

 void addVertex(T vertex);
 boolean hasVertex(T vertex);
 void addEdge(T fromVertex, T toVertex);
 UnboundedQueueInterface<T> getToVertices(T vertex);
 void clearMarks();
 void markVertex(T vertex);
 boolean isMarked(T vertex);
 T getUnmarked();
}

public GraphImplementation<T> implements GraphInterface<T> {
      .
      .
      .

public boolean hasCycleFrom(T v){




}
```

IV) Programming Question

Provide an implementation for isSorted for a generic ArrayList. You can assume that T implements Comparable<T>.

```
public boolean isSorted(ArrayList<T> arrayList) {
```