CMSC132, Practice Questions

Notice the final exam can include material not covered by the practice questions. You should practice beyond what is covered in this document. Although solutions will not be provided, you are welcome to discuss your solutions with TAs or instructors.

Problem 1 (Software Development & Testing)

1. What is the main reason many software projects fail?
    a. Poorly trained programmers
    b. Insufficient project funding
    c. Complexity of projects
    d. Slow computers
    e. Insufficient computer memory

2. What is the software life cycle?

3. What is the first phase of the software life cycle?
    a. Testing
    b. Coding
    c. Design
    d. Specification
    e. Documentation

4. True or False
    According to the waterfall model…

    a. Design all algorithms before coding
    b. Write test cases before coding
    c. Use prototype implementation to refine design

5. True or False
    According to the unified model…

    a. Design all algorithms before coding
    b. Write test cases before coding
    c. Use prototype implementation to refine design

6. True or False
    Compared to program verification, empirical testing…

    a. Handles larger programs
    b. Always catches more errors
    c. Ensures code is correct
    d. Can be applied without examining code

7. Given the following problem description, produce an object-oriented solution.

   Design a simulation of a basketball conference. Each conference has 10 teams. Each team has 12 players. Each player has a specific height, speed, and accuracy. Players know which team they belong to. Some players are scholarship players. Scholarship players need to record their current grade-point average. Players may be transferred between teams. Teams play basketball games against other teams in the conference. The result of each game is determined using a function based on the height, strength, speed, and accuracy of the players on each team.

   a. What are the objects in your object-oriented solution?
   b. What are the interactions between your objects?
   c. Which objects "have" other objects? (Also list target object)
   d. Which objects "use" other objects? (Also list target object)
   e. Which objects "are" other objects? (Also list target object)

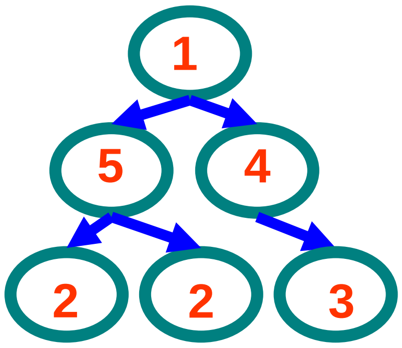
Problem 2 (Trees)

1. Binary search trees (BST)
   a. What is the key property of a binary search tree?
   b. On average, what is the complexity of doing an insertion in a binary search tree?
   c. On average, what is the complexity of doing a find in a binary search tree?
   d. What is the worst-case complexity of doing a find in a binary search tree?
   e. What can cause worst-case behavior in a binary search tree?

2. Binary search trees examples
   a. Draw the binary search tree created when the following values are inserted in order: 6, 5, 2, 8, 10, 3
   b. Given the previous BST, draw the tree created when the node 5 is removed

3. Traversals
   a. What is a tree traversal?
   b. What is the difference between a depth-first and breadth-first traversal?
   c. Pre-order, in-order, and post-order are all depth-first traversals T or F
   d. Pre-order traversals are faster than post-order traversals T or F
   e. For the following binary tree, provide an example of a

      i. Preorder traversal
      ii. Postorder traversal
      iii. Breadth-first traversal



4. Given the following Java class definition for a binary tree

```
public class BinarySearchTree <K extends Comparable<K>, V> {
   private class Node {
      private K key;
      private V data;
      private Node left, right;

      public Node(K key, V data) {
         this.key = key;
         this.data = data;
      }
   }
   private Node root;

}
```

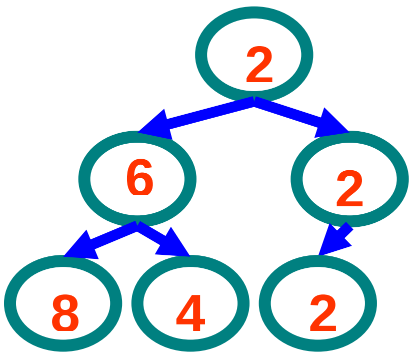
Write the following code:

a. find(K key) – Use a recursive algorithm to find a node with a key value that corresponds to key in tree
b. treeHeight( ) – Use a recursive algorithm to find the height of a tree
c. preorderPrint( ) – Use a recursive algorithm to print the key and data for each node using a preorder traversal of the tree, starting from the root.
d. subMap(K lower, K upper) – Use a recursive algorithm to implement a method that returns a BinarySearchTree with elements that have a key value between lower (inclusive) and upper (inclusive).
e. retrieveLeaves(Map<K,V> map) – Use a recursive algorithm to implement a method that initializes the map with elements that are leaf nodes. The leaves must be removed from the original tree.
f. size() – Implement the size method without using recursion.
g. thread() – Add an additional Node reference (e.g., Node thread) to the BinarySearchTree and implement a method that threads the tree based on different criteria. For example, given a tree, the thread() method can set the thread reference of every node to point to the next element in an inorder traversal.

Problem 3 (Heaps)

1. Properties & characteristics
   a. What are two key properties of a heap?
   b. What operation(s) supported by binary search trees are not supported by heaps?
   c. On average, what is the complexity of doing an insertion in a heap?
   d. On average, what is the complexity of doing a find in a heap?
   e. How can heaps be stored in a compact array representation?
   f. Why are heaps used to implement priority queues?

2. Examples
   a. Draw the heap created when the following values are inserted in order: 6, 5, 2, 8, 10, 3
   b. Given the previous heap, draw the heap produced when the smallest value is removed
   c. Show the tree representation of the following heap (in array representation) A, B, C, D, E, F
   d. Show the array representation of the following heap (in tree representation)



Problem 4 (Algorithmic Complexity)

1. Algorithmic complexity
    a. What is algorithmic complexity?
    b. List a reason benchmarking is better than analyzing complexity
    c. What is the difference between best case, worst case, and average case?
    d. What does the Big-O notation represent?

2. Finding critical regions

Calculate the asymptotic complexity of the code snippets below (using Big-O notation) with respect to the problem size n:

a.

```
for (i = 1; i < n; i = i * 2) {
     for (j = 1; j < n; j++) {

               ...

     }

}
```

f(n) = O(   )

b.

```
for (i = 0; i < n - 2; i++) {
     for (j = 0; j < n; j = j * 2) {

          for (k = 1; k < 5000; k = k * 5) {

               ...

          }

     }

     for (j = 0; j < 1000 * n; j = j + ) {

               ...

     }

     for (j = 0; j < n / 5; j = j + 5) {

               ...

     }

}
```

f(n) = O(   )

Problem 5 (Graphs)

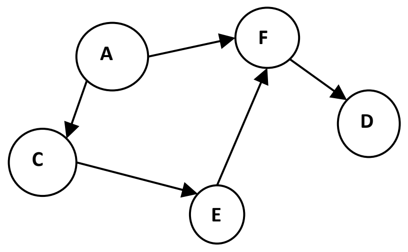
1. Properties
    a. Describe the difference between a directed and undirected graph
    b. Describe the difference between a path and a cycle
    c. Describe two methods of storing edges in a graph. Which requires more space?

2. Traversals
    a. Why is graph traversal more difficult than a tree traversal?
    b. Describe the difference between a breadth-first and depth-first traversal of a graph
    c. Given the following Java class definition for a graph

```
public class MyGraph<E> {
   public class Node<E> {
      E myValue;
       boolean tag;
       ArrayList<Node> myNeighbors;
   }
   ArrayList<Node> myNodes;
   void visitNode(Node n)       {/* Action to be performed when traversing node */}
   void deptFirstSearch(Node n) { /* Perform depth-first search of graph starting at n */ }
}
```

i. Write code for the method depthFirstSearch(n) that performs a depth first traversal starting at node n.
ii. Write code for the method breadthFirstSearch(n) that performs a breadth first traversal starting at node n.

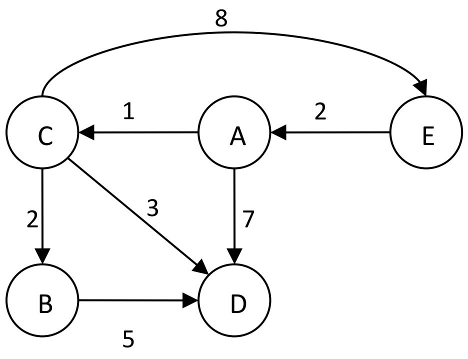
3. Graph traversals



a. Give *two* different possible orders in which the nodes of this graph could be visited in performing a **breadth-first search (BFS)** starting at node **A**. Note: there may be more than two possible orders that BFS could visit the nodes; you only have to give two orders.

b. Give *two* different possible orders in which the nodes of this graph could be visited in performing a **depth-first search (DFS)** starting at node **A**. Note: there may be more than two possible orders that DFS could visit the nodes; you only have to give two orders.

4. Single source shortest paths



Apply Dijkstra's algorithm using **A** as the starting (source) node. Indicate the cost and predecessor for each node in the graph after processing two nodes (**A** and another node). Remember to update the appropriate table entries after processing a node (after it has been added to the set of processed nodes). An empty table entry implies an infinite cost or no predecessor.

After processing the first node. Write the name of the node you are processing here: ____A____

Node	A	B	C	D	E
Cost					
Predecessor					

After processing the second node. Write the name of the node you are processing here: ________

Node	A	B	C	D	E
Cost					
Predecessor					

Problem 6 (Java Language Features)

1. Java Inner Classes
   a. What are inner classes?
   b. What are nested classes?
   c. When should inner classes be used?
   d. When should anonymous inner classes be used?
   e. What relationship exists between lambda expressions and anonymous inner class?
   f. Write an example anonymous inner class in Java.

2. Java support for Object-Oriented programming

a.	All non-static initialization blocks are executed when objects are created	T or F
b.	Code in initialization blocks are executed at the end of every constructor	T or F
c.	If no visibility modifier is specified, methods are private by default	T or F
d.	Protected access is less visible than package access	T or F

3. Generics
   a. What are the advantages of using generics?
   b. Define a class that implements a double-ended queue (dequeue) for integer values. Modify the class so it becomes a generic class.

4. Lambda Expressions
   a. What is a lambda expression?
   b. Write a lambda expression for the Runnable interface. The run() method will print even values between 1 and 100.
   c. What property must an interface have in order to be used with a lambda expression?
   d. Provide a listener (e.g., for a JavaFX button) that relies on an inner class, an anonymous inner class, and a lambda expression.

5. Exceptions in Java
   a. What are exceptions?
   b. How are exceptions used in Java?
   c. What should be made an exception in Java?
   d. What are the differences between try, catch, and finally in Java?
   e. What is the difference between checked and unchecked exceptions?
   f. Given the following code

```
public static void f( int y ) {
   try {
      System.out.print("A");
      int x = 1 / y ;     // generates ArithmeticException if y == 0
      System.out.print("B");
   }
   catch (ArithmeticException e) {
      System.out.print("C");
   }
   finally {
      System.out.print("D");
   }
}
```

   What will be printed for the following method calls?

   1. f(1)
   2. f(0)

Problem 7 (Multithreading & Synchronization)

1. Multithreading
    a. What is the motivation for writing multithreaded Java code?
    b. What are possible states for Java threads?
    c. What is the effect of invoking the start() method of the Thread class?
    d. What is the effect of invoking the join() method of the Thread class?
    e. What is scheduling?
    f. What is the difference between preemptive and non-preemptive scheduling?
    g. What are data races?
    h. Write a Java program that can experience a data race
    i. Why should programs avoid data races?

2. Synchronization & deadlocks
    a. What is synchronization?
    b. Why should programs use synchronization?
    c. What are Java locks?
    d. How may Java locks be used?
    e. What are deadlocks?
    f. Write a Java program that can experience deadlock.

3. Multithreading code

    Consider the following code:

```
public class mySet {
   List myElements = new ArrayList( );

   public boolean add( Object o ) {
      myElements.add( o );
   }

   public Object remove( ) {
      if (myElements.isEmpty( ) == false)
         return myElements.remove( 0 );   // removes & returns object at position 0
      return null;
   }
}
```

    a. What may happen if an object of the class mySet is used by multiple threads calling add( ) and remove( ) at the same time?
    b. Change the add( ) and remove( ) methods so that the class mySet can be safely used by multiple threads at once.

4. Implement a Java program that increases the value of each entry of a two-dimensional array using threads. For example, each row can be processed by a different thread. Once all the entries have been increased, the program will print the sum of all the updated entries.

5. How can you use threads to speed up the processing of a search operation in a binary search tree?

Problem 8 (Graphic User Interfaces)

1. In a GUI, what is the model? The view? The controller?
2. Why should they be kept separate?
3. What are events?
4. Why are events used in GUIs?

Problem 9 (Sorting & Algorithm Strategies)

1. Sorting algorithms
   a. What is a comparison sort?
   b. When is a sorting algorithm not a comparison sort?
   c. What is a stable sort?
   d. What is an in-place sort?
   e. What is an external sort?
   f. What is the average case complexity of sorting using
      i. bubble sort
      ii. heap sort
      iii. quick sort
      iv. counting sort
   g. What is the worst case complexity of sorting using
      i. selection sort
      ii. tree sort
      iii. heap sort
      iv. radix sort
   h. Can the following sort be performed in a stable manner?
      i. bubble sort
      ii. quick sort
      iii. counting sort
   i. Can the following sort be performed using an in-place algorithm?
      i. selection sort
      ii. tree sort
      iii. merge sort

2. Algorithm strategies
   a. What is divide-and-conquer?
   b. What is dynamic programming?
   c. What is the difference between divide-and-conquer and dynamic programming?
   d. What is the difference between recursive and backtracking algorithms?
   e. What is the difference between a greedy algorithm and heuristics?
   f. What is the difference between brute force and branch-and-bound algorithms?
   g. List a reason to use dynamic programming
   h. List a reason to use backtracking
   i. List a reason to use a brute force algorithm
   j. What type of algorithm is Kruskal's algorithm for finding minimum spanning trees?
   k. A new graph algorithm can find the best set of nodes that solve a problem by selecting each time the node that have the lowest number of neighbors until no more nodes are left. Which algorithm strategy is this algorithm relying on?
   l. Which algorithm strategy (in addition to recursion) can be used to efficiently compute Fibonacci numbers? Select only one.

Problem 10 (Hashing)

1. What are the advantages of hashing?
2. Describe the Java Hash Code Contract.
3. A class has as instance variables a string, an integer, a float and a StringBuffer object reference. Define possible hashCode methods for this class.
4. What would happen if you use a HashMap or HashSet with a class that does not implement the Java Hash Code Contract?

Problem 11 (Design Patterns)

1. Design patterns
   a. What is a design pattern?
   b. When are design patterns used?
   c. List 3 types of design patterns. Give 2 example patterns for each type.
   d. Write a Java example of the Singleton pattern
   e. Write a Java example of the State pattern
   f. List 2 examples of design patterns used in the Java class libraries
   g. Based on the following code, use the Decorator design pattern to
      i. Add a BoatDecorator class implementing the Boat interface
      ii. Create two BoatDecorators withBarnacle( ) and withTurboEngine( ) that change the result returned by topSpeed( ) by –1 and +10, respectively
      iii. Use BoatDecorators to create a Boat object for a SpeedBoat with 2 barnacles and 1 turbo engine whose topSpeed( ) method returns 48.

```
public interface Boat {
   int maxCapacity;
   int topSpeed( )
}

class CruiseShip implements Boat {     // big boat
   int topSpeed( ) { return 20; }
}

class SpeedBoat implements Boat {             // small boat
   int topSpeed( ) { return 40; }
}

Boat myBigBoat = BoatFactory.create("big");
Boat mySmallBoat = BoatFactory.create("small");

public class BoatFactory {
   static Boat create(String s) {
     // your code here
   }
}
```

Problem 12 (Linear Data Structures)

1. Name three kinds of linked lists traversals we saw in class.
2. Name one advantage of a linked list over an array list.
3. Given the following Java class definition for a linked list

```
public class LinkedList<T extends Comparable<T>> {
   private class Node {
      private T data;
      private Node next, thread;

      private Node(T data) {
         this.data = data;
         next = thread = null;
      }
   }

   private Node head, tail;
   public LinkedList() {
      head = tail = null;
   }
}
```

   Write the following code:

   a. find(K key) – Use a recursive algorithm to find a node with a key value that corresponds to key in the list
   b. printListReverse – Use a recursive algorithm to print the list in reverse order
   c. removeEvenNumberedNodes – Use a recursive algorithm to remove even-numbered nodes from a list. Re-implement your code using a non-recursive algorithm
   d. removeDuplicates() – Use a recursive algorithm to remove duplicates from the list
   e. removeInRange(T lower, T upper) – Use a recursive algorithm to remove nodes present in the range defined by lower (inclusive) and upper (inclusive). Elements removed must be placed in a set