CMSC132
Final Exam Practice Questions

Problem 1 Software Engineering & Object Oriented Design

A. Software Development and Testing
   a. Software is difficult because programmers are slow                     T or F
   b. Software life cycle refers to how software is used                  T or F
   c. Problem specification is a component of software development       T or F
   d. Problem specification is less important than program testing       T or F
   e. Iterative development is a software development methodology        T or F
   f. Black box testing is usually easier than clear box testing         T or F
   g. Integration tests are usually more important than unit tests       T or F
   h. Test coverage includes consideration of lines of code tested       T or F
   i. Test drivers are only found on NASCAR training tracks             T or F

B. Object-oriented design
   a. State, behavior, and identity are the main qualities of objects     T or F
   b. Object oriented design produces faster programs                    T or F
   c. List two main principles of object-oriented design?                T or F
   d. Inheritance describes a relationship between classes                T or F
   e. Inheritance discourages code reuse                                   T or F
   f. Extension is a form of inheritance                                  T or F

C. Object-oriented design II. Given the following problem description, produce an object-oriented solution. Answer the following questions about your 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)
   f. Draw a UML diagram of your solution.

Problem 2 Algorithmic Complexity

D. 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?

e. Why are \( O(n^2) \) and \( O(n) \) not considered equivalent?

E. 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) {
    f(n) = \( O(\quad ) \)
    for (j = 1; j < n; j++) {
        ...
    }
}

b. 
for (i =1; i < n-2; i++) {
    f(n) = \( O(\quad ) \)
    for (j = 1; j < n; j=j*2) {
        for (k = 1; k < 5000; k=k*5) {
            ...
        }
    }
    for (j =1; j < 1000*n; j=j+1) {
        ...
    }
    for (j = 1; j < n/5; j=j+5) {
        ...
    }
}

Problem 3 Data Structures and Recursion

F. Taxonomy & properties

a. Describe the main difference between linear and hierarchical data structures

b. What is the key property of a binary search tree?

c. On average, what is the complexity of doing an insertion in a binary search tree?

d. Pre-order, in-order, and post-order are all depth-first traversals T or F

e. What operation(s) supported by binary search trees are not supported by heaps?

f. What is the difference between a set and a map?

g. What happens when an open addressing hash table is close to full?

h. Describe the 2 main parts of a recursive algorithm
Problem 4 Graph Algorithms

G. Properties
   a. Describe the main difference between hierarchical and graph data structures
   b. Describe the difference between a directed and undirected graph
   c. Describe the difference between a path and a cycle
   d. Describe two methods of storing edges in a graph. Which requires more space?

H. 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 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.

I. Minimum spanning trees
   a. What is a spanning tree?
   b. Describe Kruskal’s algorithm for finding minimum spanning trees
   c. Describe Prim’s algorithm for finding minimum spanning trees
   d. Describe two methods for finding connected subgraphs
   e. Consider the following graph. Using both Prim’s and sKruskal’s algorithm, calculate the minimum spanning tree, listing edges in the minimal spanning tree in the order they are added to the tree.
J. Single source shortest path
   a. Describe Dijkstra’s algorithm for finding shortest paths in a graph
   b. Consider the previous graph. Apply Dijkstra’s algorithm for this graph to calculate the shortest path from S to every other node. Store intermediate results in the table BestKnownDistances. Show the entries in the table after you finish computing the shortest distance from S to nodes in the set {S, A, B}.

   \[ \text{Table BestKnownDistances} \]

<table>
<thead>
<tr>
<th>S</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>LowestCost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Predecessor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Order Added</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

c. Which node would be processed next using Dijkstra’s algorithm?
d. Update the table BestKnownDistances after adding this node.
e. Using Dijkstra’s algorithm, calculate the shortest path (and its cost) from S to every other vertex in the graph. List vertices in the order they are added to the table BestKnownDistances.

Problem 5 Compression & Huffman Codes

K. Compression
   a. What are two sources of compressibility?
b. What are two types of compression?
c. What is a prefix code?
d. What are Huffman codes?
e. How do Huffman codes achieve compression?
f. Given the following Huffman tree, decode the sequence “00110010”

   ![Huffman Tree](image-url)
g. Using the same Huffman tree, encode the string “arts” as a sequence of 0’s and 1’s
h. Given the following symbol frequencies, create a Huffman tree for the symbols
   A = 5, B = 8, C = 4, D = 6, E = 7

**Problem 6 Java Language Features**

L. 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. Write an example anonymous inner class in Java.

M. Java support for Object-Oriented programming
   a. The equals() method in Java is typically used to test for name equivalence  T or F
   b. All non-static initialization blocks are executed when objects are created  T or F
   c. Code in initialization blocks are executed at the end of every constructor  T or F
   d. If no visibility modifier is specified, methods are private by default  T or F
   e. Protected access is less visible than package access  T or F

N. 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

```java
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)

O. (4 pts) Cloning and serialization in Java
   a. What is cloning?
   b. What is the relationship between clone( ) and the == operator?
   c. What is the relationship between clone( ) and equals( )?
   d. What is serialization?
   e. What is serialization used for?
   f. What is the difference between making a shallow copy versus making a deep copy?

Problem 7 Multithreading & Synchronization (20 pts)

P. 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?

Q. 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. Why should programs avoid deadlocks?
   g. Write a Java program that can experience deadlock.

R. 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.
c. Change the add( ) and remove( ) methods so that the method remove( ) will always return an object when used by multiple threads (by waiting until an object has been added).

Problem 8 Networking & Networking Support in Java

S. Networking
   a. What are protocols?
   b. What is the internet?
   c. What are packets?
   d. What is IP?  UDP?  TCP/IP?
   e. What is a socket?  Port?  URL?
   f. What is the difference between reliable and unreliable network connections?
   g. How can a reliable connection be built on top of an unreliable network?
   h. What is a server?  Client?
   i. What is the difference between Java Socket, ServerSocket, and DatagramSocket?
   j. How is data transported across a Java Socket?  Across a DatagramSocket?

Problem 9 Graphic User Interfaces

T. GUIs and MVC
   a. In a GUI, what is the model?  The view?  The controller?
   b. Why should they be kept separate?
   c. What are events?
   d. Why are events used in GUIs?
   e. How are events handled in the Java Swing library?

Problem 10 Sorting & Algorithm Strategies

U. 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
V. 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?

Problem 11 Design Patterns

W. Design patterns
   a. What is a design pattern?
   b. How were design patterns discovered?
   c. When are design patterns used?
   d. List 5 components of a design pattern
   e. List 3 types of design patterns. Give 2 example patterns for each type.
   f. What type of design pattern is the factory pattern? Visitor pattern?
   g. Write a Java example of the Singleton pattern
   h. Write a Java example of the Factory pattern
   i. Write a Java example of the Visitor pattern.
   j. List 2 examples of design patterns used in the Java class libraries
   k. Given the following code, complete the code for a BoatFactory class so it can be used to create big and small boat objects:

   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 {

   }
```java
static Boat create(String s) {
    // your code here
}

1. Using the same 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.

Problem 12 Effective Java

X. Effective Java
   a. Give 3 examples of possibly confusing Java features.
   b. Write an example of potentially confusing Java code.
   c. Name 2 approaches to Java programming styles that avoids confusing Java features.
   d. Give an example of potentially confusing Java code, and how to avoid it

Problem 13 Advanced Tree Structures (Honors Section Only)

Y. Indexed search trees
   a. What is the motivation for using an indexed search tree (trie)?
   b. What is a compressed trie?
   c. What is a compact trie?
   d. What is a suffix trie?
   e. Draw the suffix trie for the string “google”
Z. (4 pts) Balanced search trees
   a. What is the motivation for using balanced search trees?
   b. Name two algorithms for maintaining balanced search trees
   c. What is the mechanism used to balance search trees?
   d. Given the following binary search tree, draw the tree resulting from performing a
      single right rotation around the node X
   e. For the same search tree, draw the tree resulting from performing a single left rotation
      around the node X

AA. (4 pts) Multi-way search trees
   a. What is the motivation for using multi-way search trees?
   b. What is a 2-3 tree?
   c. Describe the algorithm for finding items in a 2-3 tree
   d. Given the following 2-3 tree, draw the tree resulting from inserting the value 11.
   e. Given the same 2-3 tree, draw the tree resulting from inserting the value 22.