Fall 2011 Software Engineering Qualifying Exam

This is an open book exam. Basic calculators are allowed, but no computers or other devices that have communication capability are allowed; this means that cell phones cannot be used as calculators. There are a total of 100 points on this exam. Be sure to show your work in case your answer may deserve partial credit, but do not add spurious or frivolous content in hopes that something you say might be right. Content in an answer that is irrelevant to the problem may cause point deductions.

[40pts] 1. UML and Modeling

For the parts in this section, use the following problem description.

We need to build an integrated information and inventory management system for an automobile parts store. The store stocks and sells auto parts. To be able to keep track of parts (and find them for customers), the system also needs information about automobiles. Automobiles are classified into various types: trucks, sedans, and hatchbacks. Trucks have both a weight and volume capacity, sedans have a trunk volume capacity, and hatchbacks have a cargo capacity. All vehicles have a manufacturer, a model name and number, and a year. All vehicles are made up of many components, all of which we call parts. Some parts are atomic (indivisible) parts, while others are composite parts that are themselves made up of parts. The store sells both kinds of parts. Vehicles can also have optional components. Parts have a manufacturer, a part number, a name and description. Information from the manufacturer on a part also includes a list of equivalent parts. For example, a Delphi VR321 headlight might be equivalent to a GMC RE91 headlight.

One of the functionalities that the system will have to support is a customer kiosk, where a customer can walk in, click on or enter their vehicle make, model, etc., and zoom down through composite components to get to the part they need, then pull up a list of all equivalent parts, their prices, and their availability (how many are in stock in the store).

[25pts] A. Draw a problem-domain UML class diagram for this problem, including classes and their attributes, and associations between classes. Where appropriate be sure to indicate name, direction, and cardinality for associations.

[5pts] B. Would building this system be a high-risk or a low-risk project? Explain your answer.

[5pts] C. Are their any design patterns you would use in building this project? Explain your answer.

[5pts] D. What risks would their be in adding a customer web interface to the system, in addition to the in-store kiosk? Explain your answer, including what you would do to minimize those risks.

[60pts] 2. Program Analysis and Verification

The following questions deal with the program shown below, and with the sample execution shown below the program. The class is meant to implement a hash table, and the sample main() program should print out each of the command line arguments with an "item: " in front of it, and nothing else.

```
01:
02: public class HashTable
03: {
04: private final int MAX_ELEMS = 20;
05: private int /*! spec_public !*/ numElements;
06: private int /*! spec_public !*/ nextI;
07: private Object myData[];
08:
09: public HashTable()
10: {
11:    numElements = 0;
12:    nextI = -1;
13:    myData = new Object[MAX_ELEMS];
14: }
15:
16: private int hash(String s)
17: {
18:    int h=0;
19:    int i=0;
20:    while (i < s.length()) {
21:       h += s.charAt(i++);
22:    }
23:    return h % MAX_ELEMS;
24: }
25:
26: public void insert(Object val)
27: {
28:    int index = hash(val.toString());
29:    while (myData[index] != null)
30:       index = (index+1) % MAX_ELEMS;
31:    myData[index] = val;
32:    numElements++;
33: }
34:
35: public void start()
36: {
37:    nextI = 0;
38: }
39:
40: public boolean hasNext()
41: {
42:    if (nextI >= 0) return true;
43:    return false;
44: }
45:
46: public Object next()
47: {
48:    while (nextI < MAX_ELEMS && myData[nextI]==null)
49:       nextI++;
50:    if (nextI >= MAX_ELEMS) {
51:       nextI = -1;
52:       return null;
53:    }
54:    return myData[nextI++];
55: }
56:
57: public static void main(String args[])
58: {
59:    int i=0;
60:    HashTable h = new HashTable();
61:    while (i < args.length) {
62:       h.insert(args[i++]);
63:    }
64:    h.start();
65:    while (h.hasNext()) {
66:       System.out.println("item: " + h.next());
67:    }
68: }
69:
70: };
71:
72:
```

A sample run of the program is:

```
shell> java HashTable hello there and goodbye
item: goodbye
item: and
item: hello
item: there
item: null
```

[15pts] A. Write JML *requires* and *assures* clauses for the *insert()* and *next()* methods, using your knowledge of how a hash table such as the one implemented ought to behave (they may or may not be implemented correctly). In your JML annotations you should capture the ``intent'' of the operation.

[5pts] B. The execution above shows that there is an error in the class; yet it is incorrect to state that *next()* should never return null. Under what conditions should it return null? Under what conditions should it never return null?

[8pts] C. If you can, write JML annotations that capture your answer to (B). If you cannot, explain what is required that is hard or impossible to write within JML's framework.

[5pts] D. A coverage tool tells us that the execution above did not execute line 30. What type of test case would you have to construct to achieve coverage of line 30?

[10pts] E. For the object field variable *nextI*, write down all of the definitions and all of the uses, using line numbers (add a letter for multiple definitions or uses on one line). Then for only the definitions that occur in the *next()* method, write down every def-use pair and classify it into one of the scopes defined in the paper "Dataflow Testing of Classes". The scopes are: intra-method, inter-method, and intra-class.

[7pts] F. Which of the def-use pairs do you know were NOT exercised in the execution shown above? Would it be possible to provide a test case to this program that would exercise them? Would it be possible to write a program that used this class (or to simply re-write *main()*) that exercised them?

[10pts] G. Define and describe soundness and completeness in the context of program verification. The tool ESCJava claimed that it was neither sound nor complete. Give examples of why it fails on both properties. Finally, if the tool is not sound nor complete, why is it valuable at all?