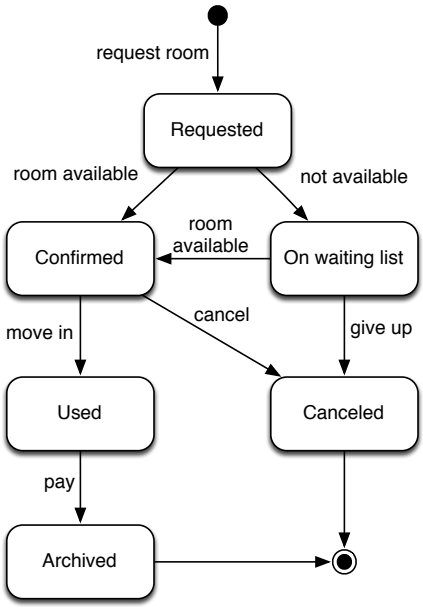
Final Exam

Tuesday 21 December 2010

Choose four out of the five questions. You have two hours, if you need it. Write your answers on separate sheets of paper, with your name on each sheet and the problem number clearly labeled. You may not use books, notes, computers, or other devices. You may leave when you have completed the exam.

1. (Requirements) Examine the state/transition diagram below and answer the following questions.



(a) According to the diagram, is every confirmed hotel room eventually paid for? Explain.

(b) Is every request eventually confirmed? Explain.

(c) Suppose this is an expensive hotel that hosts presidents and other heads of state. Every guest must pass a background check before they can move in; those that fail are kept 'waiting' indefinitely. Change the state diagram to reflect this new policy.

2. (Design) Recall that *cohesion* is a measure of how well the various responsibilities and methods of a class fit together. A class with poor cohesion can usually be refactored into multiple classes.

   Data cohesion is one kind of cohesion, where we evaluate what data members each method accesses, to see what the methods have in common.

   The following class has two data members and five methods. Analyze the data cohesion of the class, and recommend a possible way to refactor it.

```
1  class BankAccount
2  {
3      private int balance;
4      private String owner;
5
6      public void withdraw(int amount) {
7          balance = balance - amount;
8      }
9
10     public void deposit(int amount) {
11         withdraw( - amount );
12     }
13
14     public void getOwner() {
15         return owner;
16     }
17
18     public void setOwner(String newOwner) {
19         sendNotification(owner);
20         sendNotification(newOwner);
21         owner = newOwner;
22     }
23
24     public void sendNotification(String recipient) {
25         // ...
26     }
27 }
```

3. (Implementation) Briefly describe how the *branching* and *merging* features of a version control system can be helpful to software developers.

4. (Verification) As a test engineer for Boeing, you have been assigned to test the following pseudo-code. It has two parameters, `altitude` and `pitch`.[1]

```
FUNCTION autoPilot (altitude, pitch : integer)
BEGIN
    IF altitude > 10000 AND pitch < 0
    THEN RETURN pitch / 2
    ELSE IF altitude > 5000 AND pitch < 70
         THEN RETURN pitch * 0.3
         ELSE IF pitch < -25
              THEN RETURN (- pitch) / 2
              ELSE RETURN altitude + pitch
END
```

   Your colleague suggests the following three test cases. Each case contains proposed inputs (values for `altitude` and `pitch`) and the expected output (return value).

altitude	pitch	return
10384	70	10454
986	–32	16
768	0	768

   Do these three test cases produce good *coverage*? If not, suggest one or two additional cases that will make your testing more comprehensive. (Be specific: give proposed inputs and expected output.)

5. (Maintenance) The first law of software evolution[2] states that "a program that is used undergoes continual change or becomes progressively less useful." Do you agree with this statement? It's not obviously true – if the program was useful at one time, why would it not continue to be useful into the future? Explain, and give concrete examples of systems for which you think the law is and is not true.

[1] It doesn't matter to this problem, but *altitude* is the distance of a plane off of the ground, and *pitch* is the angle of the nose with respect to the ground.

[2] formulated in a 1974 article by Lehman and Belady