This assignment asks you to write some code and solve some pen-and-paper problems. You should submit the solutions to the pen-and-paper problems in class as a hard copy assignment. We will arrange a demo for the coding component of the exercise.

The first programming problem will require you to use the Spin model checker. Please download this tool from [http://spinroot.com](http://spinroot.com) and go through the online tutorials before you attempt the problem.

1. **(Pen-and-paper)**. Show that the following pairs of LTL formulas are not equivalent. You can do this by constructing an example path that satisfies one of them but not the other:
   (a) ♦□p and □(p → Xp)
   (b) ♦□p and (¬p U □p)

2. **(Pen-and-paper)**. Express in LTL the following properties (we assume that s_0s_1s_2... represents an arbitrary path):
   (a) If p occurs at least twice, then p occurs infinitely often.
   (b) If p holds at a state s_i, then q must hold at at least one of the two states just before s_i — i.e., s_{i-1} and s_{i-2}.
   (c) p never holds at less than two consecutive states. That is, if p holds at a state s_i, then it holds either at the state s_{i+1} or at the state s_{i-1}.

3. **(Pen-and-paper)**. Explain in English the properties expressed by the following LTL formulas:
   (a) ♦□p
   (b) p U ¬p
   (c) ¬p U □p.

4. **(The frog-pond problem; programming)**. Consider the frog pond shown in Figure 1. Three female frogs are on the three stones on the left and three male frogs are on the three stones on the right. Find a way to exchange the positions of the male and female frogs, so that the male frogs are all on the left and the females are all on the right. (You may first want to try it online at: [http://www.hellam.net/maths2000/frogs.html](http://www.hellam.net/maths2000/frogs.html)). The constraints that your solution must satisfy are as follows: frogs can only jump in the direction they are facing. They can either jump one rock forward if the next rock is empty or they can jump over a frog if the next rock has a frog on it and the rock after it is empty.
You can find definitions of possibly unfamiliar symbols, like '_', online at: http://spinroot.com/spin/Man/promela.html

(b) (10 pts) Does your answer change if the channel declaration is replaced with:

```
chan q = [0] of { mtype, byte, bit };
```

Explain your answer.

3. (3 pts) Three missionaries and three cannibals come to a river and find a boat that holds two. If cannibals outnumber missionaries on either side of the river, the missionaries will be eaten. How can all six get safely across the river without any one being eaten?

Use Spin to model the possibilities and claim that the problem is unsolvable. Show both the Spin model and the solution you found. Keep the model simple.

4. (3 pts) Consider the frog pond shown in Figure 1.

Three female frogs are on the three stones on the left and three male frogs are on the three stones on the right. Find a way to exchange the positions of the male and female frogs, so that the male frogs are all on the left and the females are all on the right. (You may first want to try it online at: http://www.hellam.net/maths2000/frogs.html)

The constraints that your solution must satisfy are as follows:

- Frogs can only jump in the direction they are facing.
- They can either jump one rock forward if the next rock is empty or they can jump over a frog if the next rock has a frog on it and the rock after it is empty.

Figure 1 The Frog Pond Puzzle

(a) Model the above system using in a Spin model, and show that it is possible to reach the desired end state. (Hint: you can, but need not, avoid unnecessary action interleavings by using atomic sequences to define complete hops — you should allow only one frog to hop at a time.) Show the model and explain the solution found.

(b) Consider a generalization of the problem to \( n \) male frogs and \( m \) female frogs, and use Spin to either find a solution, or to prove that no solution exists, for a few different values of \( m \) and \( n \).

5. (Bounded model checking; programming) Your goal is to extend the bounded model checker from Assignment 1 so that it can find violations of certain liveness properties.

- Extend the checker so that it can find violations of the property \( (p \lor q) \), where \( p \) and \( q \) are atomic propositions.
- Extend the checker so that it can find violations of the property \( \Box \Diamond p \) — that is, \( p \) occurs infinitely often. Once again, \( p \) is an atomic proposition.

In each case, you should write a few test cases that you can use to establish, during the demo, that your tool works properly.