CSci 117 Lab 13

1. **Missionaries and Cannibals.** This classic river-crossing AI problem consists of three missionaries, three cannibals, and one boat. The dynamics of the problem are simple: the boat can contain at least one and at most two people, and for each side of the river the missionaries cannot be fewer than the cannibals (otherwise …). Find a single solution to this problem using relational programming. Depth first search will lead to an infinite loop so you will be using breadth first search. This may still explore to many choices, so you may need to prune some of the states. In this case, find a solution to the problem, and determine what choices are needed. This problem can be structured as a set of trips back and forth across the bank, with two terminating conditions, outnumbered missionaries(fail), or everyone on crossing the river(success). It is your job to come up with a state representation, and the possible choices that can be made before each trip.

2. **Graph Coloring.** A graph is represented as a set of vertices, and edges between the vertices. Find a coloring of the graph, where no two adjacent vertices (vertices with an edge connecting them), share the same color. Vertices are integers, and edges are pairs v1#v2 connecting to vertices. Colors can be {blue, green, red}. Find a single solution for the graph with vertices [1 2 3 4 5 6], and edges [1#2 1#3 2#4 2#5 2#6 3#4 5#6] (note that edges are bidirectional). Then, create a graph that will not have a solution, and test it to see if it fails.

3. **Summation over unary natural numbers**. Numbers are written as z (zero) or s(N) (successor of the number N). So to write 3, you would have s(s(s(z))). Of course, this can be represented in Oz with nested records. The following rule scheme defines recursively the operation sum on natural numbers in a prolog style.

```
sum z N2 N2.
sum (s N1) N2 (s N3) :- sum N1 N2 N3.
```

Write this operation as a three argument procedure, where you have the ability to decide the inputs and outputs, similar to the append function.

4. **Parsing**. Update the parser from Figure 9.5, 9.6 in the following ways: Use the IntransVerb senior, TransVerb knows, Noun student, and add two more names. Write the new grammar as a comment, and update the parser to match this new grammar. Do several tests. Think about how you could include ‘and’ or ‘or’ into the grammar, and into the parser.