CMSC330 Fall 2013 Quiz #4 Solutions

1. (8 pts) Prolog

   Given the following clauses, list all answers returned by the following queries.

<table>
<tr>
<td>

```
hobbit(frodo).
hobbit(samwise).
human(aragorn).
human(gandolf).
taller(gandolf,aragorn).
taller(X,Y) :- human(X),hobbit(Y).
```

a. (2 pts) ?- human(Z).
Z=aragorn;
Z=gandolf.

b. (2 pts) ?- taller(gandolf,Z).
Z=aragorn;
Z=frodo;
Z=samwise.

</td>
<td>

```
foo([ ],X,[X]).
foo([H|T],X,[H|R]) :- foo(T,X,R).
```

c. (2 pts) ?- foo([ ],2,X).
X=[2].

d. (2 pts) ?- foo([a,b,c],d,X)
X=[a,b,c,d].

</td>
</tr>
</table>

2. (12 pts) Multithreading

Consider the following multithreaded Ruby code. Assume there are multiple threads being executed in the program, but just a *single* Market object. If we freeze program execution at some point in time, each thread *T* will have just finished executing some statement *S* (i.e., statement *S* will have been the last statement executed by thread *T*). We wish to examine the state of these threads.

```
class Market
  def initialize
    @count = 0
    @myL = Monitor.new
    @myC = @myL.new_cond
  end
```

```
  def produce
    @myL.synchronize {
1.    @myC.wait_while(@count >9)
2.    @count += 1
3.    @myC.broadcast
    }
  end
```

```
  def acquire
    @myL.synchronize {
4.    @myC.wait_while(@count<2)
5.    @count = @count- 2
6.    @myC.broadcast
    }
  end
end
```

a. (3 pts) Is it possible given two threads x and y for the last statement executed by both threads to be statement 4 in the code above? Explain your answer.

Yes, since both x & y may be consumers waiting for market to have 2 items for purchase. Whichever thread reached statement 4 first releases the lock after calling wait_while, so the other thread may acquire the lock and also execute statement 4.

b. (3 pts) Is it possible given two threads x and y for the last statement executed by thread x to be statement 4, and the last statement executed by thread y to be statement 2? Explain your answer.

Yes, since both x may be a consumer waiting for market to have 2 items for purchase. X reaches statement 4 first and releases the lock after calling wait_while, so the producer thread Y may acquire the lock and execute statements 1 & 2.

c. (3 pts) Is it possible given two threads x and y for the last statement executed by thread x to be statement 5, and the last statement executed by thread y to be statement 2? Explain your answer.

No, since if the last statement executed by a thread is either statement 2 or 5, it must be holding the lock. No other thread can acquire the lock and reach statement 2 or 5.

d. (3 pts) Is it possible to modify the code above to reduce how often threads are woken up, but must go right back to sleep because there is no work to perform? Explain your answer.

Yes. Both waiting producer and consumer threads are being awoken whenever broadcast is called, since they are all waiting on the same condition variable. The code may be modified to have two condition variables, one for producers and one for consumers. Then producers may be woken when the market is not full, and consumers may be awoken when there is sufficient products to purchase.