# CSC-363: Problem Set 7: Prolog II

**1.** Let `L` be a list of unique integers. Moreover, assume `L` contains an *odd* number of values. Assume we want to define a Prolog predicate `median(L,M)` that is true when `M` is the median of `L`. One way to implement this predicate is to sort `L` and then extract the "middle" element of the sorted list. This however is a bit of overkill.

A simpler implementation can be extracted from the definition of what it means for `M` to be `L`'s median: `M` must occur in `L` and if `M` is used to partition `L` into two sublists, one containing values greater than `M` and the other containing values smaller than `M`, then the resulting sublists must be equal in length.

Use this definition to implement `median(L,M)`. (Remember, you do not have to tell Prolog how to find `M`. Just define the relationship between `M` and `L` and let Prolog do the searching.)

Initials: _____

**2.** Recall that in languages like Scheme, ML and Prolog, sets can be represented as lists. However, unlike lists, the order of values in a set is not significant. Thus both `[1,2,3]` and `[3,2,1]` represent the same set.

(a) Write facts and rules that define a Prolog relation `setEq(S1,S2)` that tests whether sets `S1` and `S2` (represented as lists) are equal. Two sets are equal if they contain exactly the same members, ignoring ordering. In this part you may assume that sets contain only atomic values (numbers, and symbols). You may also assume that `S1` and `S2` are ground (that is, they are constants or are bound to fixed values). For example,

```
setEq([1,2,3], [3,2,1]). ⇒ yes
setEq([1,2], [3,2,1]). ⇒ no
setEq([curly,larry,moe], [moe,larry,curly]). ⇒ yes
```

Initials: _____

(b) In general, sets can contain other sets. Extend your solution to part (a) to allow sets to contain other sets. For example,

```
setEq([1,[2,3]], [[3,2],1]). ⇒ yes
setEq([1,2,3], [[3,2],1]). ⇒ no
setEq([1,2,3], [[1,2,3]]). ⇒ no
```

Initials: _____