CSC-363: Problem Set 6: Prolog I

1. Define rules for the logical expressions below. Make sure to test thoroughly.

- `and(A, B)` -- Returns true if and only if both `A` and `B` are true.
- `or(A, B)` -- Returns true if `A` and `B` are true.
- `equal(A, B)` -- Returns true if `A` and `B` are the same value.
- `xor(A, B)` -- Returns true if and only if one of `A` and `B` are true.
- `nor(A, B)` -- Returns true if `A` and `B` are not true.
- `nand(A, B)` -- Returns true if and only if both `A` and `B` are not true.

Note: `not(A)` is defined in the prolog language. Also, `false` is not defined in Prolog; however, `fail` can be substituted.

2. Define rules for the functionality below. Make sure to test thoroughly.

- Write a rule that will return the first element of a list. Call it `head`.
- Write a rule called `addToFront` that will append an element to the beginning of a list.
- Write a series of rules that will return the last element of a list. Call it `last`.
- Write a series of rules that will return the second-to-last element of a list. Call it `second_to_last`.
- Find the number of elements in a list. Call it `length`.

3. Define rules for the functionality below. Make sure to test thoroughly.

- Duplicate the elements of a list (e.g., `explode([1, 2, 3])` → `[1, 1, 2, 2, 3, 3]`). Call it `explode`.
- Write a series of rules that creates a list containing all integers within a given range.
- Reverse a list. Call it `reverse`.
- Determine whether a list is a palindrome. Call it `palindrome`.
- Eliminate consecutive duplicates of list elements. Call it `compress`. (e.g., `compress([1, 1, 2, 2, 3, 4, 3])` → `[1, 2, 3, 4, 3]`).
- Write a series of rules that will return the `K`'th element of a list. Call it `element_at`.