Name: Rev. Joseph Graydon Leveille IV

Student ID: 3955537

Ground rules:

* No person is fair game, besides yourself (and question clarification queries to the instructor, obviously… for the benefit of you pesky would-be lawyers.)
* A generally Google-able (as in, not a person over google chat/nice try) resource is fair game.
* You must do 2 of the challenges. You may do 3 for extra credit (but please do not jeopardize a grade on another exam trying to do this). Please remove the “unused” challenge from your turn in if you only do 2.

First Challenge, Choose one of 3?:

// For one of {Lisp, Clojure, Scala} code the following:

// We’re going to help out the Matlab fans out there who think elements are numbered from 1….

// and build a compatibility function for list removal, by….

// removing the X-1 element from a list.

// If there is no X-1 element, handle the error appropriately.  
//  
// Example:  
// removeAt(2, List('a, 'b, 'c, 'd))

// should “actually” remove c. But Matlab would have it that it would be “b”.

// so your should return a c d

// you are free to return the results/outputs in the way easiest for you to manage this challenge

// in terms of data structure or answer style, etc.

**Please cut and paste your code into a document and provide a screen shot of it running.**

(defn matlabRemove [fullList normAddress]

(def matAddress (- normAddress 1))

(def newList (list\* nil))

(if (> matAddress 0)

(if (> (count fullList ) matAddress)

(filter (fn [x] (not (= x (nth fullList matAddress))))

fullList)

(println "Out of Bounds! #1")

)

(println "Out of Bounds! #2")

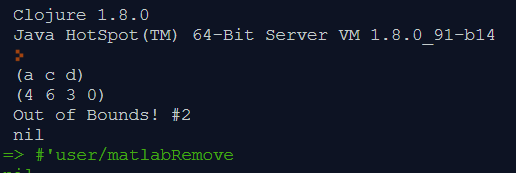
)

)

(println (matlabRemove '(a b c d) 2))

(println (matlabRemove '(4 6 2 3 0) 3))

(println (matlabRemove '(a b c d) 0))



Second Challenge, Prolog:

// Encode a *Classifier Tool* for something you know.

// Likely places you left your keys (always a problem for me)

// If someone is a member of a particular sort of student org

// (think…. likesPuzzles…. Chess club or Math Club? for example)

// Or, which building someone spends the most time in (major(X) / knowsPerson(Y)).

// plus at least 2 rules that chain the facts (at least 6). Anything pretty much goes here….

// EXCEPT: No Animals. We’ve done that to death.

// Tip - try <https://www.metalevel.at/prolog/expertsystems>

**Please cut and paste your code into a document and provide a screen shot of it running.**

Code:

major(jonah,me). major(jackson, physics).

major(jenny,ee). major(terry, cs).

major(joe,cpe). major(vidhi,be).

major(tommy, english).

engineering(X) :- major(X, cpe);major(X, me);major(X, ee);major(X, be);major(X, ce).

gellersonite(\_x) :- engineering(\_x); major(\_x, cs); \_x = jackson.

artAndScience(\_x) :- not(engineering(\_x)); gellersonite(\_x), major(\_x, cs).

knowsRosasco(\_x) :- gellersonite(\_x).

Input:

major(jenny, ce).

major(jenny, ee).

gellersonite(jackson).

artAndScience(jackson).

engineering(jackson).

engineering(joe).

engineering(tommy).

artAndScience(tommy).

knowsRosasco(X).

;

;

;

;

;

;

Output:

