CS378 - A Formal Model of the JVM
Lecture 2

http://www.cs.utexas.edu/users/moore/classes/cs378-jvm/

Semester	Spring, 2012
Unique Id	53105
Intructor	J Strother Moore
Email	moore@cs.utexas.edu
Office	MAI 2014
Office Hours	MW 1:00–2:00

Cons versus List

```
(cons 1
      (cons (+ 2 3)
            (cons (* 2 3)
                  nil)))
```

can also be written

```
(list 1 (+ 2 3) (* 2 3)).
```

Elements versus Nodes

There are 3 elements in:

```
(cons 1
      (cons (cons 2 3)
            (cons 4
                  nil)))
```

aka

```
(list 1 (cons 2 3) 4)
```

Represent Stacks

Pick a representation for stacks. Define `push` so that `(push i stk)` pushes the item `i` onto the stack represented by `stk`.

Define `top` to take a non-empty stack and return the topmost item.

Define `pop` to take a non-empty stack and return the stack obtained by removing the topmost item.

Programming Note

The symbols `push` and `pop` are already defined in the standard ACL2 symbol package.

To define these functions in an ACL2 session we have to create a new package, which we'll call the "M1" package.

I'll show you the incantation for that later.

Accessing List Elements

Define `nth` so that `(nth i x)` returns the $\texttt{i}^{th}$ (0-based) element of list `x`. You may assume `x` has at least `i`+1 elements.

Updating List Elements

Define `update-nth` so that
`(update-nth i v x)`
"changes" the list `x` so that the `i`th (0-based) element is `v`. It leaves the other elements unchanged. Actually, it returns a new list; you can't modify an object in ACL2. You may assume `x` has at least `i`+1 elements.

M1 Package

```
(defpkg "M1"
 '(T NIL QUOTE IF EQUAL AND OR
   NOT IMPLIES IFF CONS CAR CDR CONSP ENDP
   LIST LIST* ATOM SYMBOLP + - * / EXPT
   FLOOR MOD NATP INTEGERP NFIX ZP < <=
   > >= LET LET* COND CASE OTHERWISE DEFUN
   DEFTHM THM DEFCONST DEFMACRO PROGN &REST
   MUTUAL-RECURSION IN-PACKAGE DECLARE
   IGNORE XARGS IN-THEORY ENABLE DISABLE
   E/D INCLUDE-BOOK LD I-AM-HERE PBT PCB
   PCB! PE PE! PF PL PR PR! PUFF U UBT UBT!
   O-P O< ACL2-COUNT INTERN-IN-PACKAGE-OF-SYMBOL
   COERCE SYMBOL-NAME STRING CONCATENATE
   STRIP-CARS ASSOC PAIRLIS$ PAIRLIS-X2
   SYNTAXP QUOTEP))

(in-package "M1")
```

```
(defpkg "M1"
 '(T NIL QUOTE IF EQUAL AND OR
   NOT IMPLIES IFF CONS CAR CDR CONSP ENDP
   LIST LIST* ATOM SYMBOLP + - * / EXPT
   FLOOR MOD NATP INTEGERP NFIX ZP < <=
   > >= LET LET* COND CASE OTHERWISE DEFUN
   DEFTHM THM DEFCONST DEFMACRO PROGN &REST
   MUTUAL-RECURSION IN-PACKAGE DECLARE
   IGNORE XARGS IN-THEORY ENABLE DISABLE
   E/D INCLUDE-BOOK LD I-AM-HERE PBT PCB
   PCB! PE PE! PF PL PR PR! PUFF U UBT UBT!
   O-P O< ACL2-COUNT INTERN-IN-PACKAGE-OF-SYMBOL
   COERCE SYMBOL-NAME STRING CONCATENATE
   STRIP-CARS ASSOC PAIRLIS$ PAIRLIS-X2
   SYNTAXP QUOTEP))

(in-package "M1")
```

Note that `PUSH` (i.e., `M1::PUSH`) is undefined in this package!

```
(defpkg "M1"
 '(T NIL QUOTE IF EQUAL AND OR
   NOT IMPLIES IFF CONS CAR CDR CONSP ENDP
   LIST LIST* ATOM SYMBOLP + - * / EXPT
   FLOOR MOD NATP INTEGERP NFIX ZP < <=
   > >= LET LET* COND CASE OTHERWISE DEFUN
   DEFTHM THM DEFCONST DEFMACRO PROGN &REST
   MUTUAL-RECURSION IN-PACKAGE DECLARE
   IGNORE XARGS IN-THEORY ENABLE DISABLE
   E/D INCLUDE-BOOK LD I-AM-HERE PBT PCB
   PCB! PE PE! PF PL PR PR! PUFF U UBT UBT!
   O-P O< ACL2-COUNT INTERN-IN-PACKAGE-OF-SYMBOL
   COERCE SYMBOL-NAME STRING CONCATENATE
   STRIP-CARS ASSOC PAIRLIS$ PAIRLIS-X2
   SYNTAXP QUOTEP))

(in-package "M1")
```

The only ACL2 functions you can use are those listed above!