CS3304 Principles of Programming Languages Arthur

Lisp Programming Assignment

Due: 12/2/14

Scenario: You are to write a lisp program that a robot can use to get from point A to point B in a deterministic (non backtracking) way.

Your program, navigate, will have two input parameters:

(1) a destination represented as an integer, and
(2) a "map" that represents all paths the robot can follow.

The "map" is in the form of a binary search tree. Each node in the tree is of the form

(position (left subtree) (right subtree)).

Because the paths are represented as a BINARY SEARCH TREE, comparing the given destination to the "current" position will enable you to determine whether you want to turn right (dest < curr pos) or left (dest > curr pos) to get to your destination, if it exists.

Your program will print out

(1) (not found: #), if the destination "#" does not exist,

(2) (found: #), if the destination is the root node (# is root node number),

(3) (found: # T # T ... # T #), if your destination is in the tree but not the root (# is a position number and T is either L or R indicating that you took a Left or Right turn at position #.

Suppose, for example, the set of possible paths looked like the following

```
                         ROBOT
                           V
           Right Turn               Left Turn
                           9
        +------------------^------------------+
        |                                     |
        7                                     26
 +------^------+                      +-------^-------+
 |             |                      |               |
()             8                     20               31
         +-----^------+          +---^----+       +---^---+
         |            |          |        |       |       |
        ()            ()        17        ()     ()       ()
                            +---^---+
                            |       |
                           ()       ()
```

The corresponding Sexpression given to navigate as the path argument would be:

```
(9 (7 () (8 () ()))
   (26 (20 (17 () ())
           ())
       (31 () ())))
```

(navigate 17 "path")

would then return "(found: 9 L 26 R 20 R 17)"

indicating that the robot must travel to position 9, turn right, go to position 26, turn left, go to position 20, turn left, and then go to position 17, its destination. **BEWARE!!! Traveling down the LEFT subtree implies a RIGHT turn, and vice-versa.**

(navigate 9 "path")

would return "(found: 9)"

indicating that the robot could proceed directly to position 9 and that it is actually the root of the binary search tree.

(navigate 30 "path")

would return "(not found: 30)"

indicating that the given destination could not be found.

You will upload a tar file to Scholar containing

(a) the scheme file having both the program definitions and invocations, and

(b) a listing of the output generated by invoking NAVIGATE on a data set that I will provide 2 days before the program is due.

You will also hand in paper copies of both (a) and (b) above during class on the date due.

You are restricted to:

The LISP Numeric Primitives: +, -, *, /

The LISP Predicates: =, <, >, >=, <=, <>, even?, odd? and zero?

symbol, number, list? and null?

The LISP functions: define, quote ('), car, cdr, cons, cond, eq?, and equal?

The car and cdr family of functions, e.g., caar, cadr, cddr, etc.

If you need additional helper functions, ***define*** them.

.

You can obtain a Lisp/Scheme interpreter (Dr. Racket) from the web site noted on the Class website. When you download your Dr. Racket interpreter, *the Language you will select will be "R5RS" under Legacy Languages*. This will define how your interpreter works and to which functions you have access. Ultimately, your program must execute on RLOGIN.

Place your program definitions and invocations in the same file. I will execute your program on RLOGIN using "mzscheme < file_name" where file_name is the name of the file containing your scheme program and invocations. mzscheme (/usr/local/bin/mzscheme) is the non-GUI UNIX scheme interpreter provided by Dr. Racket. Use I/O redirection to create the output file.

MAKE SURE THAT YOU PROGRAM WORKS PROPERLY USING MZSCHEME ON RLOGIN!