

CPTS 355 - HOMEWORK 1 - SOLUTIONS

- (1) Parenthesis are required as a part of ordinary infix notation to eliminate ambiguity and also to over-ride operator precedence rules.
- (2) One of the main benefits of having parenthesis in Scheme is that the functions and special forms can have an arbitrary number of arguments where in Postscript each function is limited to a fixed number of arguments. For example in Postscript we have to write “2 3 add 4 add 5 add” but in Scheme we could write “(+ 2 3 4 5)”.
Some of the examples of the features which require parenthesis in LISP/Scheme are special forms, list’s and function calls.
- (3) The operators which affect the number of elements in the different stacks are given below.
 - Operand Stack:** Operators which affect the number of elements in the operand stack are “clear” and “dup”.
 - Graphics Context Stack:** Operators which affect the number of elements in the graphics context stack are “gsave” and “grestore”.
 - Dictionary Stack:** Operators which affect the number of elements in the dictionary stack are “begin” and “end”.
- (4) On line 7 the value of bold-faced x is 4.
False, the definition of x on line 2 does not occur in the same dictionary as the definition of f2 on line 7.
True, the definition of x on line 7 overwrites the definition of x on line 2.
The type of the value on the top of the stack when the program finishes is “string”.
The value on the top of stack is the string 41 * 3.
- (5) For each of the function definitions in the *problem 2.1, Mitchell* graphs are given below.
 - a:** Graph = $\{ \langle x, x * 5 \rangle \mid x > 1 \}$. The function is a partial function defined on all integers greater than 1 and undefined on integers less than or equal to 1.
 - b:** Graph = $\{ \langle x, 1 \rangle \mid \forall x \in Integers \}$. The function is a total function on integers.
 - c:** Graph = $\{ \langle x, 1 \rangle \mid (x = 0) \vee (x > 1 \ \&\& \ x \bmod 2 = 0) \}$. The function is a partial function defined on all even integers greater than 1 or the integer 0. It is undefined on all odd integers greater than 0 or integers less than 0.

- (6) Function definition for evalpoly is given below.
 (define (evalpoly coeffs x)
 (if (null? coeffs) 0 (+ (car coeffs) (* x (evalpoly (cdr coeffs) x))))).
- (7) The values for the expressions are listed below.
 (car (car L)) = The requested expression does not make sense. The second call to car has an argument 1 which is not of type list.
 (car (cdr L)) = (2 (3 4))
 (car (cdr (cdr L))) = (5)
 (cdr (cdr (cdr (cdr L)))) = The requested expression does not make sense. The last call to cdr has an argument () which is an error.
- (8) The function polystep is defined below.
 (define (polystep accum coeff) (+ (* accum x) coeff))
- (9) Yes, given $Halt_0$ which will tell us if a program reading no input will halt, we can produce a function that will tell us if a program, P , that reads one input halts on a given input, I . The problem says that program P begins with a read statement that assigns the input to a variable. Given P and I we can make a new program P' in which that read statement is replaced by an assignment statement initializing the variable to I . Clearly P halts on input I iff P' halts (on no input) so to determine if P halts on input I one need only ask $Halt_0$ if P' halts.

P :

x = read()
 rest of program

P' :

x = I
 rest of program