PPL assignment 2 Question 1 (theoretical part):

* 1. Primitive atomic expression: true

Non-primitive atomic expression: define x=5. x is non-primitive atomic expression.

Non-primitive compound expression:

Primitive atomic value: 5

Non-primitive atomic value:

Non-primitive compound value:

* 1. Special form is a compound expression wich is not evaluated like a regular compound expression. For example: cond.
  2. Free variable is a variable which has a reference in scope E, but is not declared in scope E.
  3. S expression is an atomic expression

2.

2.1)

Signature: empty? (x)

Type: [Any -> Boolean]

Purpose: Returns true iff the given expression is the empty list.

Pre-Conditions: True.

Test: (empty? ‘()) ⇒ #t

(empty? ‘(1 2 3)) ⇒ #f

2.2)

Signature: list? (x)

Type: [Any -> Boolean]

Purpose: Returns true iff the given expression is a list.

Pre-Conditions: True.

Test: (list? (cons 1 (cons 2 ‘()))) ⇒ #t

(list? (cons 1 2) ⇒ #f

2.3)

Signature equal-list? (x, y)

Type: [Any \* Any -> Boolean]

Purpose: Returns true iff the given two expressions are equal lists.

Pre-Conditions: True.

Test: (equal-list? ‘(1 2) ‘(1 2)) ⇒ #t

(equal-list? (cons 1 2) (cons 1 2)) ⇒ #f

2.4)

Signature: append (x, y)

Type: [List \* List -> List]

Purpose: Takes two lists and returns the first list appended with the second.

Pre-Conditions: True.

Test: (append ‘(1 2 3) ‘(4 5 6)) ⇒ ‘(1 2 3 4 5 6)

2.5)

Signature: append3 (x, y, n)

Type: [List \* List \* Number -> List]

Purpose: Takes two lists and one item, and returns the first list appended with the second and the integer.

Pre-Conditions: True.

Test: (append3 ‘(1 2 3) ‘(4 5 6) 7) ⇒ ‘(1 2 3 4 5 6 7)

2.6)

Signature: pascal (n)

Type: [Number -> List]

Purpose: Computes the nth row of Pascal's triangle.

Pre-Conditions: n >= 1

Test: (pascal 5) => '(1 4 6 4 1)