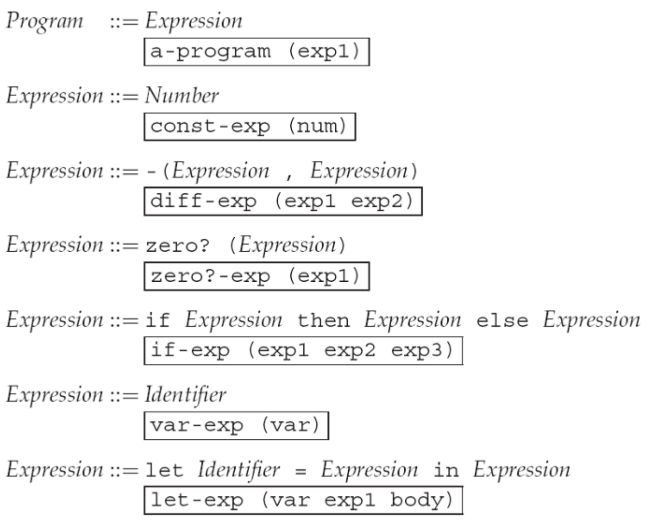
Problem Set 4
COMP301 FALL 2022
Week 4-5: 24.10.2022 - 04.11.2022

Instructions:

- Submit your answers to the Blackboard PS3 assignment until November 5 Saturday, at 23.59.
- Please submit only **one single PDF file**, where all of your codes for each of the parts are included.
- Name your submission file as *id_ username_ ps4.pdf* (Example: *00000_ yhizir19_ ps4.pdf*).



```
Program    ::= Expression
               a-program (exp1)

Expression ::= Number
               const-exp (num)

Expression ::= -(Expression , Expression)
               diff-exp (exp1 exp2)

Expression ::= zero? (Expression)
               zero?-exp (exp1)

Expression ::= if Expression then Expression else Expression
               if-exp (exp1 exp2 exp3)

Expression ::= Identifier
               var-exp (var)

Expression ::= let Identifier = Expression in Expression
               let-exp (var exp1 body)
```

Problem 1:

Part A. According to the grammar definition given above, parse the following code and write its abstract syntax tree.

```
''if zero? (x) then 55 else -(44, x)''
```

Part B. According to the grammar definition given above, unparse the abstract syntax tree given below.

```
#(struct: a-program
   #(struct: let-exp
      x
      #(struct: const-exp 97)
      #(struct: diff-exp
         #(struct: const-exp 32)
         #(struct: var-exp x))))
```

Problem 2: In the lecture you have seen a procedural implementation of the environment. Below, there is a procedural implementation of the list. Fill in the blank part.

```
(define (empty-list)
  (lambda (mode)
    (display "end␣of␣list")))

(define (prepend-list a lst)
  (lambda (mode)
    (-------------
       ----------
       ----------)))

(define (car-list lst)
  (lst #t))

(define (cdr-list lst)
  (lst #f))

; Tests:
; (define x (prepend-list 13 (prepend-list 3 (prepend-list 6
; (prepend-list 7 (empty-list))))))
; (car-list x) -> returns 13
; (car-list (cdr-list x)) -> returns 3
; (car-list (cdr-list(cdr-list(cdr-list(cdr-list x))))) -> returns "end of list"
```

Problem 3: Given a list, an element and a number, implement a procedure "remove-n-times" that removes the element from the list n times. If the element does not occurs n times in the list, it removes all occurrences.

```
(remove-n-times 'a '(a b a a b a) 2) ; returns (b a b a)
(remove-n-times 'a '(a b a) 3) ; returns (b)
```

Problem 4: Given a nested list and an input, implement a procedure named "count-occurrence-nested" that counts the occurrence of the given element in the nested list.

```
(count-occurrence-nested '(a b (a b (a b)) (a b)) 'a) ; returns 4
(count-occurrence-nested '(a (b a) (a b) (a b c)) 'b) ; returns 3
```