Assignment 4

May 28, 2023 11:40 AM

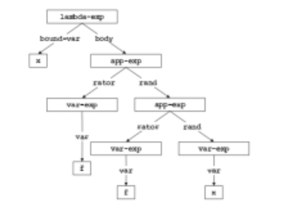
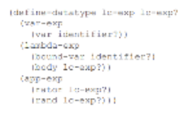


Figure 2.2 Abstract syntax tree for (lambda (x) (f (f x)))



```
(define-datatype lc-exp lc-exp?
  (var-exp
    (var identifier?))
  (lambda-exp
    (bound-var identifier?)
    (body lc-exp?))
  (app-exp
    (rator lc-exp?)
    (rand lc-exp?)))
```

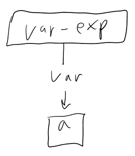
1)

a) a

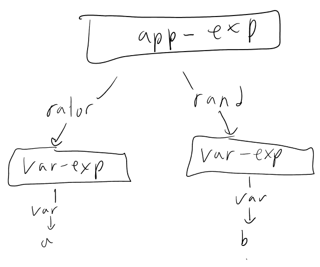
c) ((lambda (a) (a b)) c)

(app-exp (lambda-exp '(a) (app-exp (var-exp 'a) (list (var-exp 'b)))) (list (var-exp 'c)))

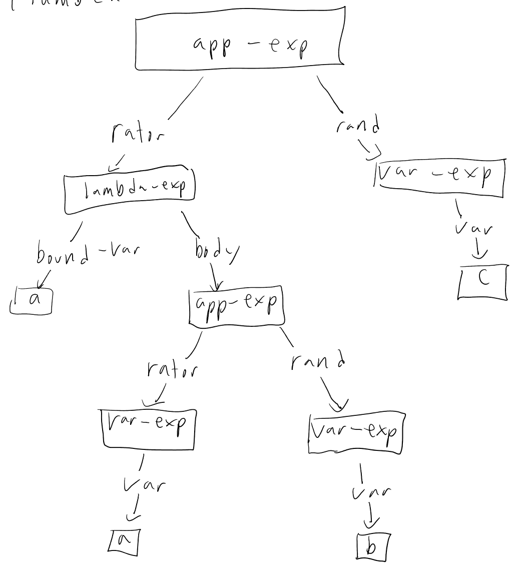
a)



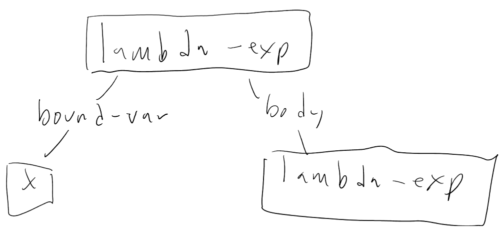
b) (a b)

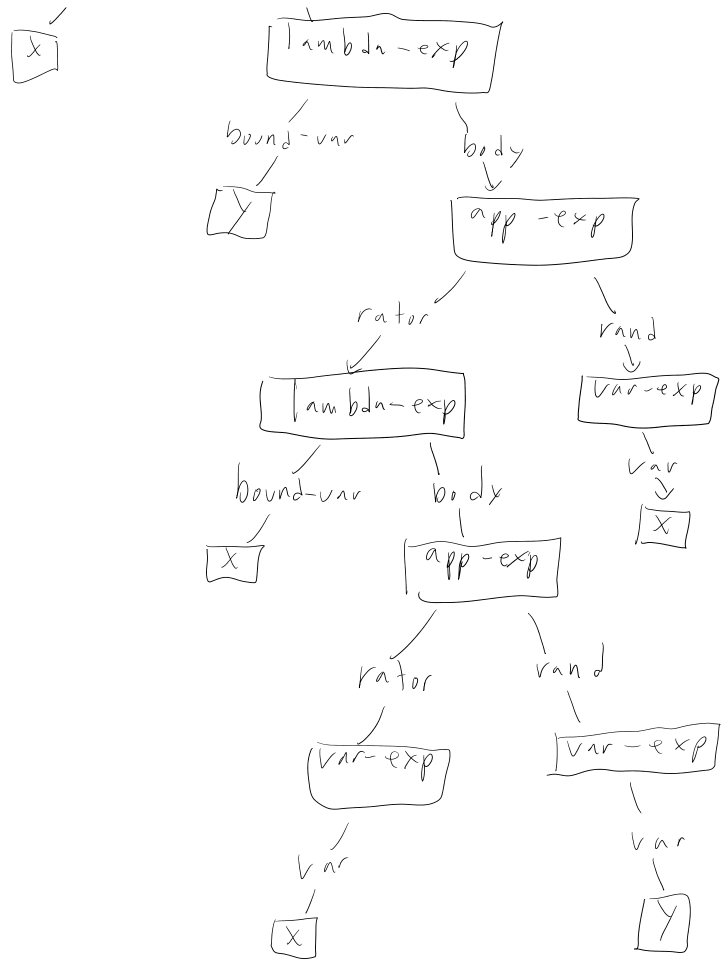


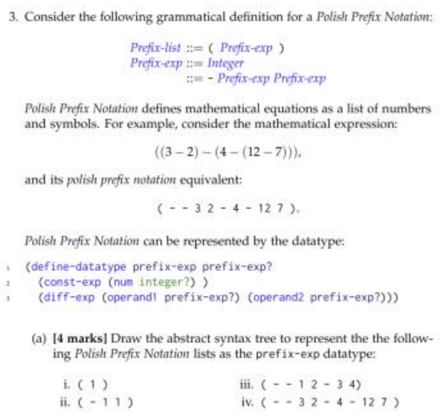
c) ((lambda (a) (a b)) c)



d) (lambda (x) (x y))







3. Consider the following grammatical definition for a *Polish Prefix Notation*:

Prefix-list ::= (*Prefix-exp*)
Prefix-exp ::= *Integer*
::= - *Prefix-exp Prefix-exp*

Polish Prefix Notation defines mathematical equations as a list of numbers and symbols. For example, consider the mathematical expression:

$$((3-2)-(4-(12-7))),$$

and its *polish prefix notation* equivalent:

`( - - 3 2 - 4 - 12 7 ).`

Polish Prefix Notation can be represented by the datatype:

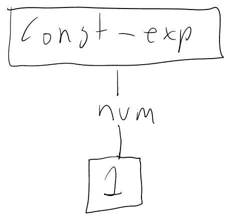
```
(define-datatype prefix-exp prefix-exp?
  (const-exp (num integer?) )
  (diff-exp (operand1 prefix-exp?) (operand2 prefix-exp?)))
```

(a) **[4 marks]** Draw the abstract syntax tree to represent the the following *Polish Prefix Notation* lists as the `prefix-exp` datatype:

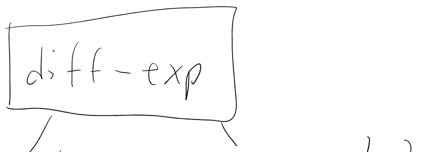
i. `( 1 )`	iii. `( - - 1 2 - 3 4)`
ii. `( - 1 1 )`	iv. `( - - 3 2 - 4 - 12 7 )`

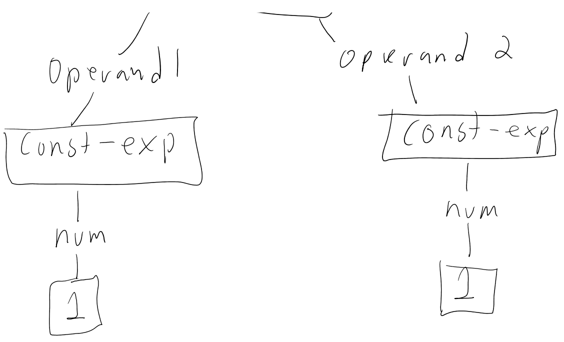
- - 3 2 - 4 - 12 7

4 a I (1)

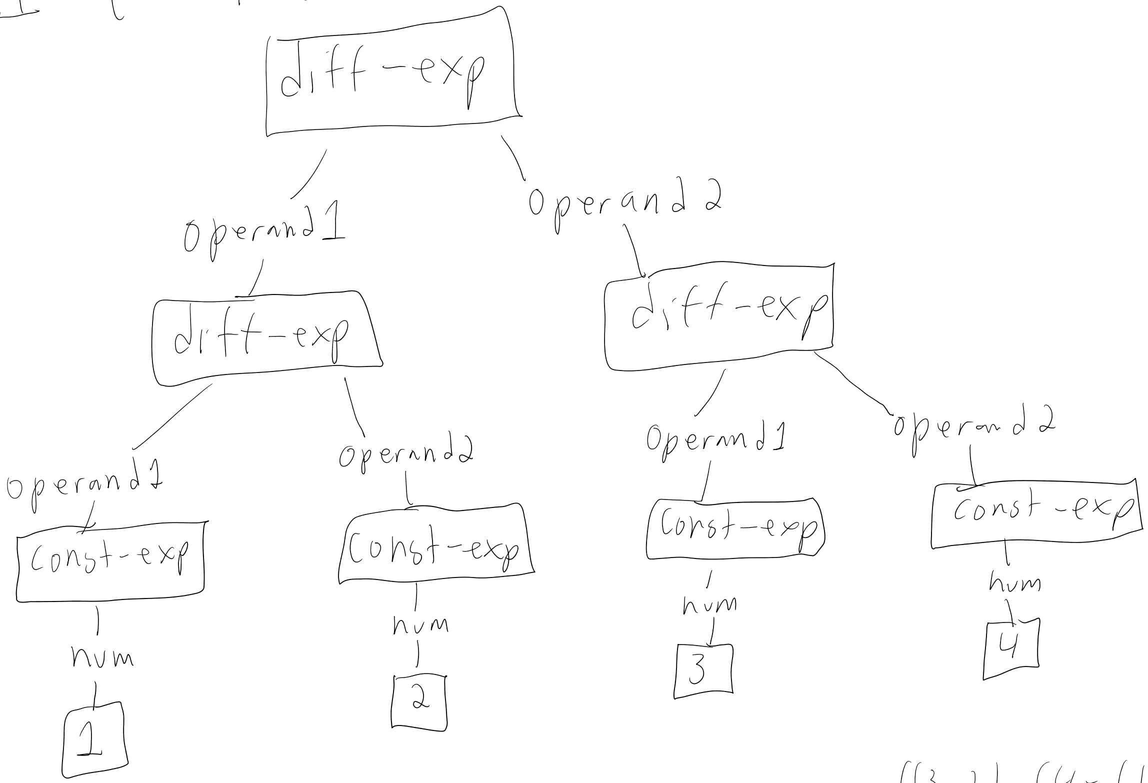


II (- 1 1)



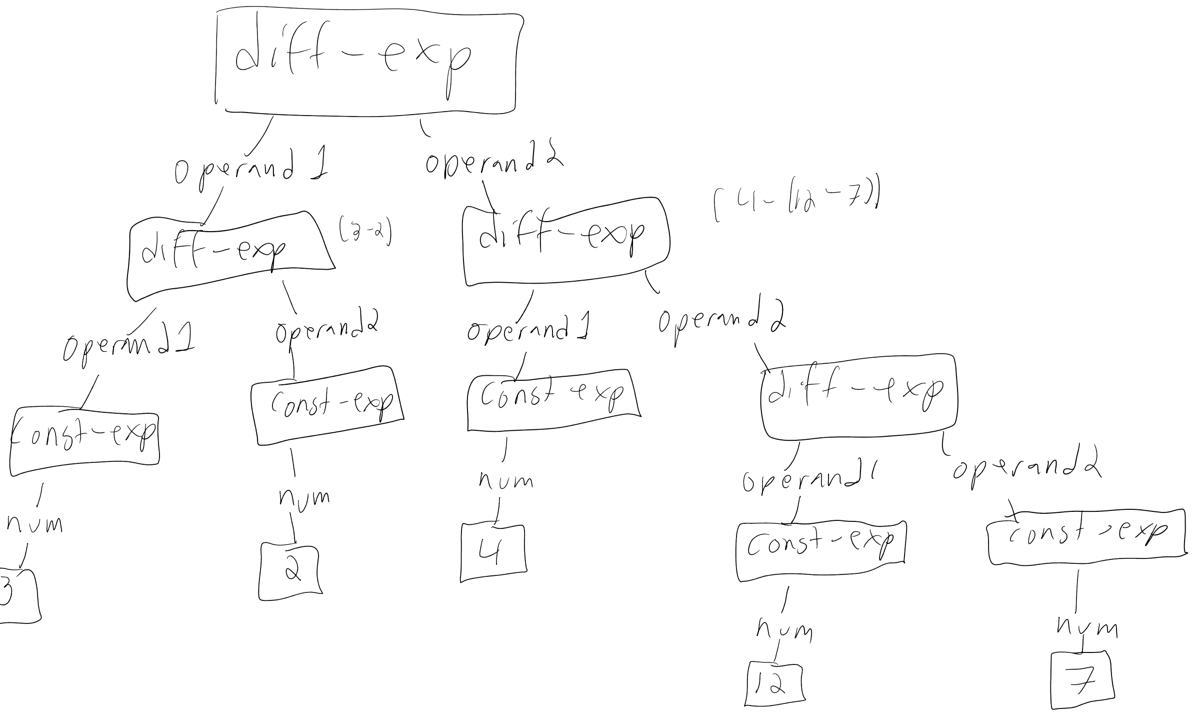


III (- - 1 2 - 3 4)



((3-2)-(4-(12-7)))

IV


diff-exp
operand 1
operand2
diff-exp
(3-2)
diff-exp
(4-(12-7))
operand1
operand2
operand 1
operand 2
Const-exp
Const-exp
Const-exp
diff-exp
operand1
operand2
Const-exp
const-exp
num
num
num
num
num
3
2
4
12
7