Homework: FuncLang (Part II) Solutions

Learning Objectives:

1. FuncLang and functional programming
2. Understand and expand FuncLang interpreter

Instructions:

- Total points: 76 pt
- Early deadline: Mar 22 (Wed) at 11:59 PM; Regular deadline: Mar 24 (Fri) at 11:59 PM (you can continue working on the homework till TA starts to grade the homework).
- Download hw5code.zip from Canvas.
- Set up the programming project following the instructions in the tutorial from Homework 2 (similar steps).
- How to submit:
  - For question 1, please submit one PDF that contains the solution.
  - For questions 2–5, please submit your solutions in one zip file with all the source code files (just zip the complete project's folder).
  - For FuncLang programs, write them in "hw5.ss" file, which can be placed inside the "examples" folder.
  - Submit the zip file and one PDF file to Canvas under Assignments, Homework 5 submission.

Questions:

1. (6 pt) [Free And Bound Variables] Identify the free and bound variables in the following two FuncLang programs.

   (a) (3 pt)

```
1  (define f1
2    (lambda (x y)
3      (if (= x y) 0
4              B B
5        (if (< x z) 2
6                B F
7          (let
8            ((f2 (lambda (a b) (+ a (* c b)))))
9                                  B    F B
10           (let
```

```
11              ((f3 (lambda (c d) (- c e)) ))
12                                   B F
13              (f3 y f2)
14                B B B
15              )
16            )
17          )
18        )
19      )
20    )
```

(b) (3 pt)

```
1  (define f4
2    (lambda (x y)
3      (if (< x y) 1
4              B B
5        (let
6          ((f4 (lambda (x z) (+ x y))))
7                                 B B
8          (< x z)
9             B B
10         )
11       )
12     )
13   )
```

2. (12 pt) [FuncLang For Nested Lists] Write FuncLang programs to process nested lists.

   (a) (4 pt) Extend the `freq` function from HW4, to act on nested lists. Given a number, n, and a nested list, made up of lists of numbers; return the frequency of n in each of the lists. See the example interaction below:

   $ (freq 7 (list (list 77 73) (list 89)))
   (0 0)
   $ (freq 77 (list (list 77 73) (list 89)))
   (1 0)
   $ (freq 66 (list (list 44 52) (list 66) (list 23 12 66) (list 66 66 66 6)))
   (0 1 1 3)
   $ (freq 77 (list (list 77 73) (list 88) (list 88 77 76) (list 88 88 88 88)))
   (1 0 1 0)
   $ (freq 7 (list))
   ()
   $ (freq 7 (list (list)))
   (0)
   $ (freq 7 (list (list) (list 7)))
   (0 1)
   $ (freq 7 (list (list) (list)))
   (0 0)

   Solution: Found in "hw5.ss" in hw5code-sol.zip.

```
1  (define freq
2    (lambda (n lst)
```

```
3     (if (null? lst)
4       (list)
5       (if (isPresent n (car lst))
6         (append (list (count n (car lst))) (freq n (cdr lst)))
7         (append (list 0) (freq n (cdr lst)))
8         )
9       )
10    )
11  )
12
13 (define isPresent
14   (lambda (n lst)
15     (if (null? lst)
16       #f
17       (if (= (car lst) n)
18         #t
19         (isPresent n (cdr lst))
20         )
21       )
22     )
23   )
24
25 (define count
26   (lambda (n lst)
27     (if (null? lst)
28       0
29       (if (= (car lst) n)
30         ( + 1 (count n (cdr lst)) )
31         (count n (cdr lst))
32         )
33       )
34     )
35   )
36
37 (define append
38   (lambda (lst1 lst2)
39     (if (null? lst1) lst2
40       (if (null? lst2) lst1
41         (cons (car lst1) (append (cdr lst1) lst2))
42         )
43       )
44     )
45   )
```

(b) (8 pt) Given a nested list, `nlst`, made up of lists of numbers; return a list that *concatenates* all the list in `nlst` and sorts the numbers into *descending* form. See the example interaction below:

$ (con-sort (list (list 1 2) (list 3 4) (list 5)))
(5 4 3 2 1)
$ (con-sort (list (list 1 2) (list 5 4) (list 5)))
(5 5 4 2 1)
$ (con-sort (list (list 1 2) (list 4 3) (list 5)))
(5 4 3 2 1)
$ (con-sort (list))
()

$ (con-sort (list (list)))
()
$ (con-sort (list (list) (list)))
()
$ (con-sort (list (list 1 2)))
(2 1)
$ (con-sort (list (list 1 2) (list)))
(2 1)
$ (con-sort (list (list) (list 1 2)))
(2 1)

Solution: Found in "hw5.ss" in hw5code-sol.zip.

```
(define append
  (lambda (lst1 lst2)
    (if (null? lst1) lst2
      (if (null? lst2) lst1
        (cons (car lst1) (append (cdr lst1) lst2))
        )
      )
    )
  )

(define concatenate
  (lambda (lst)
    (if (null? lst)
      (list)
      (if (null? (cdr (cdr lst)))
        (append (car lst) (car (cdr lst)))
        (concatenate
          (cons (append (car lst) (car (cdr lst))) (cdr (cdr lst)))
          )
        )
      )
    )
  )

(define greater
  (lambda (x y)
    (if (> x y) x y)
    )
  )

(define maximum
  (lambda (lst)
    (if (null? lst) (list)
      (if (null? (cdr lst))
        (car lst)
        (greater (car lst) (maximum (cdr lst)))
        )
      )
    )
  )

(define removefromlist
  (lambda (ele lst)
```

```
44      (if (null? lst) (list)
45        (if (= ele (car lst))
46          (cdr lst)
47          (cons (car lst) (removefromlist ele (cdr lst)))
48          )
49        )
50      )
51    )
52
53  (define sortlist
54    (lambda (lst)
55      (if (null? lst)
56        (list)
57        (let
58          ((x (maximum lst)))
59          (cons x (sortlist (removefromlist x lst)))
60          )
61        )
62      )
63    )
64
65  (define con-sort
66    (lambda (lst)
67      (if (null? lst)
68          (list)
69          (if (null? (car lst))
70              (if (null? (cdr lst))
71                  (list)
72                  (sortlist (concatenate lst))
73              )
74              (if (null? (cdr lst))
75                  (sortlist (car lst))
76                  (sortlist (concatenate lst))
77              )
78          )
79        )
80      )
81    )
```

---

3. (8 pt) [FuncLang For Strings] A built-in function, `length`, that can be used with `StringVal` to get the length of the strings is present in FuncLang. See example interaction below:

   $ (length “hello”)
   5
   $ (length “h”)
   1
   $ (length 1)
   Parameter for length was not a string.

   (a) (4 pt) Given you a list of strings, write a function `total` that report the *total length* of all the strings in the list. See the example interaction below:

   $ (total (list “hello” “hi” “you”))
   10
   $ (total (list))

0
$ (total (list “goodbye” “bye” “me”))
12

Solution: Found in “hw5.ss” in hw5code-sol.zip.

```
(define foldl
  (lambda (op zero lst)
    (if (null? lst)
      zero
      (foldl op (op (car lst) zero) (cdr lst))
      )
    )
  )

(define total
  (lambda (lt)
    (if (null? lt)
      0
      (foldl
        (lambda (x y) (+ (length x) y))
        (length (car lt))
        (cdr lt)
        )
      )
    )
  )
```

(b) (4 pt) Given you a list of strings, write a function `longest` that report the length of the longest string in the list. See the example interaction below:

$ (longest (list “hello” “hi” “you” “supercalifragilisticexpialidocious”))
34
$ (longest (list “goodbye” “bye” “me”))
7
$ (longest (list))
0
$ (longest (list “” “”))
0

Solution: Found in “hw5.ss” in hw5code-sol.zip.

```
(define foldl
  (lambda (op zero lst)
    (if (null? lst)
      zero
      (foldl op (op (car lst) zero) (cdr lst))
      )
    )
  )
(define longest
  (lambda (lt)
    (if (null? lt)
      0
      (foldl
        (lambda (x y) (if (> (length x) y) (length x) y))
        (length (car lt))
```

```
16          (cdr lt)
17          )
18        )
19      )
20    )
```

4. (20 pt) [FuncLang Interpreter] Extend the syntax and semantics of the FuncLang to add support for the logical conjunction (&&) and logical disjunction (||) expressions. See the example interaction below:

   $ (&& #t #t)
   #t
   $ (&& #t #f)
   #f
   $ (|| #t #f)
   #t
   $ (&& (< 3 4) (> 4 2))
   #t
   $(|| (> 3 4) (> 4 2))
   #t
   $ (|| 1 #f)
   Error: Expected BoolVal as first expression in (|| 1.0 #f)
   $ (|| #t 2)
   Error: Expected BoolVal as second expression in (|| #t 2.0)
   $ (&& (< 3 4) "h")
   Error: Expected BoolVal as second expression in (&& (< 3.0 4.0) "h")

   Solution: Found in hw5code-sol.zip.

   Grammar file (FuncLang.g)

```
exp returns [Exp ast]:
    va=varexp { $ast = $va.ast; }
    | num=numexp { $ast = $num.ast; }
    | str=strexp { $ast = $str.ast; }
    | bl=boolexp { $ast = $bl.ast; }
    | add=addexp { $ast = $add.ast; }
 ...
    | conj=conjexp { $ast = $conj.ast; }
    | disj=disjexp { $ast = $disj.ast; }
    ;
 ...

   conjexp returns [ConjExp ast] :
     '(' '&&'
         e1=exp
       e2=exp
     ')' { $ast = new ConjExp($e1.ast,$e2.ast); }
     ;

  disjexp returns [DisjExp ast] :
     '(' '||'
         e1=exp
```

```
23        e2=exp
24      ')' { $ast = new DisjExp($e1.ast,$e2.ast); }
25      ;
```

AST (AST.java)

```
1     public static class ConjExp extends BinaryComparator {
2     public ConjExp(Exp first_exp, Exp second_exp) {
3       super(first_exp, second_exp);
4     }
5
6     public Object accept(Visitor visitor, Env env) {
7       return visitor.visit(this, env);
8     }
9   }
10
11
12   public static class DisjExp extends BinaryComparator {
13     public DisjExp(Exp first_exp, Exp second_exp) {
14       super(first_exp, second_exp);
15     }
16
17     public Object accept(Visitor visitor, Env env) {
18       return visitor.visit(this, env);
19     }
20   }
21
22   public interface Visitor<T> {
23     // This interface should contain a signature for each concrete AST node.
24     public T visit(AST.AddExp e, Env env);
25
26 ...
27
28     public T visit(AST.ConjExp e, Env env); // Additional expressions for convenience
29
30     public T visit(AST.DisjExp e, Env env); // Additional expressions for convenience
31
32   }
```

Evaluator (Evaluator.java)

```
1  public class Evaluator implements Visitor<Value> {
2    ...
3   public Value visit(ConjExp e, Env env) { // New for funclang.
4     Value v1 = (Value) e.first_exp().accept(this, env);
5     Value v2 = (Value) e.second_exp().accept(this, env);
6     if (!(v1 instanceof BoolVal)) {
7       return new DynamicError("Error: Expected BoolVal as first expression in " + ts.
            visit(e, env));
8     }
9     if (!(v2 instanceof BoolVal)) {
10      return new DynamicError("Error: Expected BoolVal as second expression in " + ts
            .visit(e, env));
11    }
12    BoolVal bv1 = (BoolVal) v1;
13    BoolVal bv2 = (BoolVal) v2;
14    return new BoolVal(bv1.v() && bv2.v());
```

```
15   }
16
17
18   public Value visit(DisjExp e, Env env) { // New for funclang.
19     Value v1 = (Value) e.first_exp().accept(this, env);
20     Value v2 = (Value) e.second_exp().accept(this, env);
21     if (!(v1 instanceof BoolVal)) {
22       return new DynamicError("Error: Expected BoolVal as first expression in " + ts.
           visit(e, env));
23     }
24     if (!(v2 instanceof BoolVal)) {
25       return new DynamicError("Error: Expected BoolVal as second expression in " + ts
           .visit(e, env));
26     }
27     BoolVal bv1 = (BoolVal) v1;
28     BoolVal bv2 = (BoolVal) v2;
29     return new BoolVal(bv1.v() || bv2.v());
30   }
```

Printer (Printer.java)

```
1  public class Printer {
2    ...
3       public String visit(AST.ConjExp e, Env env) {
4         String result = "(&& ";
5         result += e.first_exp().accept(this, env) + " ";
6         result += e.second_exp().accept(this, env);
7         return result + ")";
8       }
9
10
11     public String visit(AST.DisjExp e, Env env) {
12       String result = "(|| ";
13       result += e.first_exp().accept(this, env) + " ";
14       result += e.second_exp().accept(this, env);
15       return result + ")";
16     }
17   }
```

5. (30 pt) [FuncLang Advanced Interpreter]

   (a) (24 pt) Extend the syntax and semantics of the FuncLang to add support for `equal?` expression, which takes two sub-expressions and returns `#t` if their values are equal and `#f` otherwise. See the example interaction below:

   $ (equal? #t #t)
   #t
   $ (equal? #t 1)
   #f
   $ (equal? #t “Hello”)
   #f
   $ (equal? (list) (list))
   #t

$ (equal? (list) (list 1))
#f
$ (equal? (list (list 1)) (list (list 0)))
#f
$ (equal? (list (list 1)) (list (list 1)))
#t
$ (equal? (cons 1 2) (cons 1 2))
#t
$ (equal? (cons 1 2) (cons 0 1))
#f
$ (equal? (+ 2 3) (+ 2 3))
#t
$ (equal? (let ((x 2)) x) (let ((y 2)) y))
#t
$ (equal? (let ((x 2)) y) (let ((y 2)) x))
funclang.Env$LookupException: No binding found for name: y

Solution: Found in hw5code-sol.zip.

Grammar file (FuncLang.g)

```
exp returns [Exp ast]:
    va=varexp { $ast = $va.ast; }
    | num=numexp { $ast = $num.ast; }
    | str=strexp { $ast = $str.ast; }
    | bl=boolexp { $ast = $bl.ast; }
    | add=addexp { $ast = $add.ast; }
 ...
    | eqs=equalsexp { $ast = $eqs.ast; }
    ;
 ...

  equalsexp returns [EqualsExp ast] :
     '(' 'equal?'
         e1=exp
       e2=exp
     ')' { $ast = new EqualsExp($e1.ast,$e2.ast); }
     ;
```

AST (AST.java)

```
  public static class EqualsExp extends BinaryComparator {
    public EqualsExp(Exp first_exp, Exp second_exp) {
      super(first_exp, second_exp);
    }

    public Object accept(Visitor visitor, Env env) {
      return visitor.visit(this, env);
    }
  }

  public interface Visitor<T> {
    // This interface should contain a signature for each concrete AST node.
    public T visit(AST.AddExp e, Env env);

```

```
15 ...
16
17     public T visit(AST.EqualsExp e, Env env); // Additional expressions for
           convenience
18
19
20   }
```

Evaluator (Evaluator.java)

```
1  public class Evaluator implements Visitor<Value> {
2    ...
3    public static boolean compareValue(Value v1, Value v2) {
4      if (v1 instanceof NumVal && v2 instanceof NumVal) {
5        NumVal first = (NumVal) v1;
6        NumVal second = (NumVal) v2;
7        return first.v() == second.v();
8      } else if (v1 instanceof StringVal && v2 instanceof StringVal) {
9        String s1 = ((StringVal)v1).v();
10       String s2 = ((StringVal)v2).v();
11       return s1.equals(s2);
12     } else if (v1 instanceof PairVal && v2 instanceof PairVal) {
13       boolean b1 = compareValue(((PairVal)v1).fst(), ((PairVal)v2).fst());
14       boolean b2 = compareValue(((PairVal)v1).snd(), ((PairVal)v2).snd());
15       return b1 && b2;
16     } else if (v1 instanceof FunVal && v2 instanceof FunVal) {
17       return v1 == v2;
18     } else if (v1 instanceof BoolVal && v2 instanceof BoolVal) {
19       return ((BoolVal)v1).v() == ((BoolVal)v2).v();
20     } else if (v1 instanceof Null && v2 instanceof Null){
21       // list
22       return true;
23     }else {
24       return false;
25     }
26   }
27
28   public Value visit(EqualsExp e, Env env) { // New for funclang.
29     Value v1 = (Value) e.first_exp().accept(this, env);
30     Value v2 = (Value) e.second_exp().accept(this, env);
31     return new BoolVal(compareValue(v1, v2));
32   }
33
34 }
```

Printer (Printer.java)

```
1  public class Printer {
2    ...
3      public String visit(AST.EqualsExp e, Env env) {
4        String result = "(equal? ";
5        result += e.first_exp().accept(this, env) + " ";
6        result += e.second_exp().accept(this, env);
7        return result + ")";
8      }
9    }
```

(b) (6 pt) Write a FuncLang program to compute the string and number compression over a list. You will identify and remove any repetitive letters or numbers in a list. See the example interaction below:

$ (compression (list “a” “b” “b”))
(“a” “b”)
$ (compression (list 1 0 0 0 0 0 1))
(1 0 1)
$ (compression (list 1 0 0 1 0 0))
(1 0 1 0)

Solution: Found in “hw5.ss” in hw5code-sol.zip.

```
(define consecutive
  (lambda (last lst)
    (if (null? lst)
      (list)
      (if (equal? last (car lst))
        (consecutive last (cdr lst))
        (cons (car lst) (consecutive (car lst) (cdr lst)))
        )
      )
    )
  )
(define compression
  (lambda (lst)
    (if (null? lst)
      (list)
      (cons (car lst) (consecutive (car lst) (cdr lst)))
    )
  )
)
```