PPL202

Assignment 2

Responsible Lecturer: Meni Adler
Responsible TA: Naama Dayan

General Instructions

Submit your answers to the theoretical questions in a pdf file called id1_id2.pdf and your code for programming questions inside the provided q2.l3, L21-ast.ts, q3.ts, q4.ts files in the correct places. ZIP those files together (including the pdf file, and only those files) into a file called id1_id2.zip. Make sure that your code abides the Design By Contract methodology.
Do not send assignment related questions by e-mail, use the forum instead. For any administrative issues (milu'im/extensions/etc) please open a request ticket in the Student Requests system.
Important: do not add any extra libraries in the supplied template files, otherwise, we will fail to compile and you will receive a grade of zero. If you find that we forgot to import necessary libraries, let us know.

Question 1: Theoretical Questions [30 points]

Q1.1 Give an example for each of the following categories in L3:

- Primitive atomic expression
- Non-primitive atomic expression
- Non-primitive compound expression
- Primitive atomic value
- Non-primitive atomic value
- Non-primitive compound value

[6 points]

Q1.2 What is a special form? Give an example [2 points]

Q1.3 What is a free variable? Give an example [2 points]

Q1.4 What is Symbolic-Expression (s-exp)? Give an example [2 points]

Q1.5 What is 'syntactic abbreviation'? Give two examples [5 points]

Q1.6 Let us define the L30 language as L3 excluding the *list* primitive operation and the literal expression for lists with items (there is still a literal expression for the empty list '()).
Is there a program in L3 which cannot be transformed to an equivalent program in L30? Explain or give a contradictory example.
[5 points]

Q1.7 In practical session 5, we dealt with two representations of primitive operations: *PrimOp* and *Closure*. List an advantage for each of the two methods [2 points].

Q1.8 In class, we implemented *map* in L3, where the given procedure is applied on the first item of the given list, then on the second item, and so on. Would another implementation which applies the procedure in the opposite order (from the last item to the first one), while keeping the original order of the items in the returned list, be equivalent? Would this be the case also for: *reduce, filter, compose* [6 points]

Answers should be submitted in file id1_id2.pdf

Question 2: Programing in L3 [25 points]

Q2.1
Implement in L3 the procedure last-element, which returns the last element of a given list.
You may assume the list is not empty.
For example:
(last-element (list 1 3 4)) → 4

Q2.2
Implement in L3 the procedure power, which given 2 numbers – n1, n2, return n1 to the power of n2 (n1^n2). You may assume the given numbers are not negative.
For example:
(power 2 4) → 16
(power 5 3) → 125

Q2.3
Implement in L3 the procedure sum-lst-power, which given a list and a number N, returns the sum of all its elements in the power of N.
For example:
(sum-lst-power (list 1 4 2) 3) → 1^3+ 4^3 + 2^3 = 73

Q2.4
Implement in L3 the procedure num-from-digits, which given a list of digits, returns the number consisted from these digits.

For example:
(num-from-digits (list 2 4 6)) → 246
(num-from-digits (list 5 7)) → 57

Q2.5
A narcissistic number is a number that is the sum of its own digits, each raised to the power of the number of the digits.
For example:

153 = 1^3+5^3+3^3=1+125+27=153
370 = 3^3+7^3+0^3=27+343+0=370
1 = 1^1=1

Implement in L3 the procedure is-narcissistic, which tests if a given number is narcissistic or not.

The procedure gets as parameter a list of digits, which represents a number.

For example:

(is-narcissistic (list 1 5 3)) → #t
(is-narcissistic (list 1 2 3)) → #f
You may add auxiliary procedures to all questions.

The code **(without comments)** should be submitted in file test/q2.l3

Don't forget to write a contract for each of the above procedures.
```
; Signature:
; Type:
; Purpose:
; Pre-conditions:
; Tests:
```
Write the contracts in file id1_id2.pdf.

You can test your code with test/q2-tests.ts

Question 3: Syntactic Transformations [25 points]

Let us define the L21 as L2 with the addition of the special form 'for', for supporting loops:
A 'for' expression is defined by: the loop variable (VarDecl), start value (NumExp), end value (NumExp), and a body composed of one expression (CExp). For example:
(for i 1 3 (* i i))
The 'for' expression is evaluated as follows: the expression in the body is evaluated for each of the loop variable values (from 'start' to 'end'), where the last evaluation is the value of the whole 'for' expression.

The evaluation of the above expression, for example, will apply (* i i) three times:
(* 1 1)
(* 2 2)
(* 3 3)
Where the last evaluation, 9, is he value of the '(for i 1 3 (* i i))' expression.

In this question you are required to implement a syntactic transformation from a given L21 AST to an equivalent L2 AST.

1. The file 'test/L21-ast.ts' contains the parser of L2. Extend this file with the new 'for' special form of L21:
   - Extend the AST type model with ForExp.
   - Complete the *parseL21* procedure, which gets an L21 program string and returns its parsed AST.

2. Implement the procedure *for2app* (at file test/q3.ts), which applies a syntactic transformation from a ForExp to an equivalent AppExp, as demonstrated in the following example:

```
(for i 1 3 (* i i))
→
(
 (lambda ()
   ( (lambda (i) (* i i)) 1 )
   ( (lambda (i) (* i i)) 2 )
   ( (lambda (i) (* i i)) 3 )
 )
)
```

3. Implement the procedure *L21ToL2* (at file test/q3.ts), which gets an L21 AST and returns an equivalent L2 AST.

Don't forget to write the contract of the *for2app,L21ToL2* procedures as a comment in the code.
<u>The code should be submitted in files test/q3.ts, test/L21-ast.ts</u>
You can test your code with test/q3-tests.ts

Question 4: Code translation [20 points]

Write the procedure *l2ToJS* which transforms a given L2 program to a JavaScript program.

The procedure gets an L2 AST and returns a string of the equivalent JavaScript program.

For example:

(+ 3 5 7) --- (3 + 5 + 7)

(= 3 (+ 1 2)) --- (3 === (1 + 2))

(if (> x 3) 4 5) --- ((x > 3) ? 4 : 5)

(lambda (x y) (* x y)) --- ((x,y) => (x * y))

((lambda (x y) (* x y)) 3 4) --- ((x,y) ⇒ (x * y))(3,4)

(define pi 3.14) --- const pi = 3.14

(define f (lambda (x y) (* x y))) --- const f = ((x,y) ⇒ (x * y))

(f 3 4) --- f(3,4)

In case the procedure body contains more than one expression, a body framed by “{}” should be defined in the translated JS code, **where each expression in the body is followed by “;”, and the last expression is prefixed by “return”**. For example:

(define g (lambda (x y) (+ x 2) (- y 3) (* x y))) --- const g = ((x,y) ⇒ {(x + 2); (y - 3); return (x * y);
});

In case the given L2 program is composed of more than one expression (, each one of them should be translated in one line, followed by an ‘;’. **The last expression in the last line should be set as a parameter of a ‘console.log’ command**:

(L2 (define b (> 3 4)) (define x 5) (define f (lambda (y) (+ x y))) (define g (lambda (y) (* x y))) (if (not b) (f 3) (g 4)) ((lambda (x) (* x x)) 7))
----->
const b = (3 > 4);
const x = 5;
const f = ((y) => (x + y));
const g = ((y) => (x * y));
((!b) ? f(3) : g(4));
console.log(((x) => (x * x))(7));

Note: As a starting point you may take the *unparse* procedure, which gets an L2 AST and returns an L2 program string. The *unparse* procedure is given in l2-ast.ts in the template files directory.

Don’t forget to write a contract for the *l2ToJS* procedure, as a comment in the code.
The code should be submitted in file test/q4.ts
You can test your code with test/q4-tests.ts

Good Luck!