Due: 2022/3/26

Homework 6 - Extend2

* If there is any problem, please contact TA.

Name:_____________ Student ID:_____________ Email: __________________

Problem 1. (40 points)
We've seen how to define natural numbers using church encoding in untyped lambda calculus:

$$\begin{aligned}
\mathbf{0} &= \lambda f.\lambda x.\ x \\
\mathbf{1} &= \lambda f.\lambda x.\ f\ x \\
&\ldots \\
\mathbf{n} &= \lambda f.\lambda x.\ f^n\ x \\
&\ldots
\end{aligned}$$

Note that church encoding cannot represent negative integers, we try to encode all integers using **untyped** lambda calculus.

(a) Propose a method to extend church numerals to representation of integers.(Hint: you may try to use pairs). Give a concrete example for representation of integer **-5** with your proposed method.

(b) Define a function *nat2int* that converts a natural number to your representation of correspondent integer.

(c) Based on this definition of integers, define the following arithmetic operations in lambda calculus(you can directly use operations on natural numbers defined before like add, multi, etc.):

  (1) negation: neg n

  (2) addition: addint m n

  (3) subtraction: subint m n

  (4) multiplication: multint m n

(d) Bonus: Are there other ways to implement integers? Explain your idea briefly with some example for operations.

Solution. For this question we directly write numbers(0,1,2,...) to represent church encoding.

(a) We can represent any integer n by a pair(a,b) and n is the difference between a and b. In other words, n=a-b. Since the integer value is more naturally represented if one of the pair is zero, we define the function *zero* to convert any pair to a pair include only one zero:

$$zero = \lambda x.iszero\ (fst\ x)\ x\ (iszero\ (snd\ x)\ x\ (zero\ (pair\ (pred\ (fst\ x))(pred\ (snd\ x)))))$$

and we use fix-point combinator to implement recursion:

$$z = \lambda fx.iszero\ (fst\ x)\ x\ (iszero\ (snd\ x)\ x\ (f\ (pair\ (pred\ (fst\ x))(pred\ (snd\ x)))))$$

$$zero = fix\ z$$

In this representation:

$$-5 = pair\ 0\ 5$$

(There are infinite pairs to encode -5 in this way, but we can always apply our function *zero* on it and get *pair* 0 5)

(b)

$$nat2int = \lambda x.pair\ x\ 0$$

(c) (1)

$$neg = \lambda x.\ pair\ (snd\ x)\ (fst\ x)$$

(2)

$$addint = \lambda m.\lambda n.\ zero\ (pair\ (add\ (fst\ m)\ (fst\ n))\ (add\ (snd\ m)\ (snd\ n)))$$

(3)

$$subint = \lambda m.\lambda n.\ zero\ (pair\ (add\ (fst\ m)\ (snd\ n))\ (add\ (snd\ m)\ (fst\ n)))$$

(4)

$$multint = \lambda m.\lambda n.\ zero\ (pair\ (add\ (multi\ (fst\ m)\ (fst\ n))\ (multi\ (snd\ m)\ (snd\ n))) \\ (add\ (multi\ (fst\ m)\ (snd\ n))\ (multi\ (snd\ m)\ (fst\ n)))$$

(d) Another way to implement integers is also using a pair(s,n), where s is the sign(tru for positive, fls for negative) and n is the absolute value. For example of -5, it can be encoded as : *pair* 0 5.

The negation is easy to define :

$$neg = \lambda x.pair\ (not\ (fst\ x))\ (snd\ x)$$

Also straightforward with multiplication:

$$not = \lambda xyz.x\ z\ y$$

$$xor = \lambda xy.x\ (not\ y)\ y$$

$$multint = \lambda mn.pair\ (xor\ (fst\ m)\ (fst\ n))\ (multi\ (snd\ m)\ (snd\ n))$$

other operations quite the same.

□

Problem 2. (30 points)
Given the definition of Fibonacci number

$$F_0 = 0, F_1 = 1, F_i = F_{i-1} + F_{i-2}$$

(a) Use *fix* to write a lambda function called *fib*: int → int to compute the n-th Fibonacci number.

(b) We want to extend simple *let* expression to recursive *let rec* expression:

$$letrec\ f = \lambda x.\ e_1\ in\ e_2$$

where f itself can appear in e_1.

Example usage of *letrec* for factorial:

$$fact = \lambda n.(letrec\ fact = (\lambda i.\ if\ i = 0\ then\ 1\ else\ i * (fact\ (i-1)))in\ fact\ n)$$

(1) Define semantic and typing rules for expression *letrec* ;

(2) Use *letrec* to redefine our Fibonacci function.

Solution.

(a)

$$\begin{aligned} ff =& \lambda f : int \to int \\ & .\lambda n : int. \\ & \quad if\ n < 2\ then\ n \\ & \quad else\ (f\ (n-1)) + (f\ (n-2)) \end{aligned}$$

$$fib = fix\ ff$$

(b) (1)

$$\frac{}{letrec\ f = \lambda x.e_1\ in\ e_2 \to e_2[(\lambda x.e_1)[letrec\ f = \lambda x.e_1\ in\ f/f]/f]}(e - letrec)$$

$$\frac{\Gamma, f : t_1 \to t_2, x : t_1 \vdash e_1 : t_2 \quad \Gamma, f : t_1 \to t_2 \vdash e_2 : t}{\Gamma \vdash letrec\ f = \lambda x.e_1\ in\ e_2 : t}$$

(2)

$$\begin{aligned} fib = \lambda n.(let\ rec\ fib = (\lambda i.\ & if\ (leq\ i\ 1)\ then\ i \\ & else\ add\ (fib\ pred\ i)\ (fib\ pred\ pred\ i)) \\ & in\ fib\ n) \end{aligned}$$

□

Problem 3. (30 points)
Given the following λ expression:

```
let x = 2 in
  let y = 4 in
    let f1 = \x.\y.x+2*y in
      let f2 = \x.\y.2*x-y in
      f2 (f1 y x) 3
```

Using the environment model for lambda calculus with let,
(a) Define closures. (Be careful and refer to lecture slides);
(b) Show detailed multi-step evaluation process of the λ expression above.

Solution.
(a) Closures:

$$C_{f1} = \{\lambda x.\lambda y.x + 2 * y, x \to 2, y \to 4\}$$
$$C_{f2} = \{\lambda x.\lambda y.2 * x - y, x \to 2, y \to 4, f1 \to C_{f1}\}$$

Don't forget to bind x and y in the closures, although they are parameters of functions.
(b) Evaluation:

(1) $(., let\ x = 2\ in\ let\ y = 4\ in\ let\ f1 = \lambda x.\lambda y.x + 2 * y\ in\ let\ f2 = \lambda x.\lambda y.2 * x - y\ in$
$f2\ (f1\ y\ x)\ 3) \to^* ...$
(2) $(x \mapsto 2, let\ y = 4\ in\ let\ f1 = \lambda x.\lambda y.x + 2 * y\ in\ let\ f2 = \lambda x.\lambda y.2 * x - y\ in\ f2\ (f1\ y\ x)\ 3) \to^* ...$
(3) $(x \mapsto 2, y \mapsto 4, let\ f1 = \lambda x.\lambda y.x + 2 * y\ in\ let\ f2 = \lambda x.\lambda y.2 * x - y\ in\ f2\ (f1\ y\ x)\ 3) \to^* ...$
(4) $(x \mapsto 2, y \mapsto 4, f1 \mapsto C_{f1}, let\ f2 = \lambda x.\lambda y.2 * x - y\ in\ f2\ (f1\ y\ x)\ 3) \to^* ...$
(5) $(x \mapsto 2, y \mapsto 4, f1 \mapsto C_{f1}, f2 \mapsto C_{f2}, f2\ (f1\ y\ x)\ 3) \to^* ...\ (Require\ (6))$
(6) $(x \mapsto 2, y \mapsto 4, f1 \mapsto C_{f1}, f2 \mapsto C_{f2}, f1\ y\ x) \to^* ...\ (Require\ (7)\ and\ (8))$
(7) $(x \mapsto 2, y \mapsto 4, f1 \mapsto C_{f1}, f2 \mapsto C_{f2}, y) \to^* 4$
(8) $(x \mapsto 2, y \mapsto 4, f1 \mapsto C_{f1}, f2 \mapsto C_{f2}, x) \to^* 2$
(9) $(x \mapsto 2, y \mapsto 4, f1 \mapsto C_{f1}, f2 \mapsto C_{f2}, f1\ 4\ 2) \to^* 8\ (Return\ (6))$
(10) $(x \mapsto 4, y \mapsto 2, f1 \mapsto C_{f1}, f2 \mapsto C_{f2}, f2\ 8\ 3) \to^* 13\ (Return\ (1))$

You can write the evaluation steps in your own way as long as it shows the environment in each step clearly! □