National University of Singapore
School of Computing
SWS3012: SICP
July 2023

R8A
Environment Model

Problems:

1. The following is the `make_withdraw` function shown in Lecture L7 and L8:

```
function make_withdraw(balance) {

    function withdraw(amount) {
        if (balance >= amount) {
            balance = balance - amount;
            return balance;
        } else {
            return "Insufficient funds";
        }
    }
    return withdraw;
}

const acc1 = make_withdraw(100);
const acc2 = make_withdraw(200);
acc1(30);
acc2(60);
```

   Draw the environment diagram for the program according to the environment model. Show all the frames that are created during the program evaluation. Show the final value of each binding.

2. Consider the following program that makes a calculator function for a final price after adding a fixed commission, considering a given tax rate. What is the result of evaluating the program? Draw the environment diagram for the program according to the environment model. Show all the frames that are created during the program evaluation. Show the final value of each binding.

```
let commission = 25; // my commission in dollars

// return a calculator for total price
// total price = (commission + cost) * (1 + tax_rate)

function make_price_calculator(tax_rate) {
    function calculator(cost) {
        return (commission + cost) * (1 + tax_rate);
    }
    return calculator;
}

const calc = make_price_calculator(0.07);
commission = 125;
calc(75);
```

3. What is the result of evaluating the following program? Draw the environment diagram for the program according to the environment model. Show all the frames that are created during the program evaluation. Show the final value of each binding.

   The environment model distinguishes between **primitive** and **pre-declared** functions, and this distinction is laid down in the Specification of Source §3—2021 edition. Calls of primitive functions do not create any frames; they simply directly produce their result. Calls of pre-declared functions create new frames.

   ```
   function curry(f) {
       return x => y => f(x, y);
   }

   (curry(math_pow))(3)(4);
   ```

   The program shows an example of the technique of *currying*, which converts a function that takes multiple arguments into a sequence of functions that each takes a single argument (https://en.wikipedia.org/wiki/Currying).