Programming Languages
CS320 Spring Semester, 2019
= Homework No. 2 =

Due: April 5, 2019 (Friday) 11:59PM (by TA’s clock)
Submit (1 rkt file) To: 周宇(Yu Zhou) by email 1263305115@qq.com

1. **`; free-ids : WAE -> list-of-sym`**

   Implement the function **`free-ids`**, which takes a **`WAE`** and produces a list of symbols. The list should contain a symbol for each free identifier in the given `WAE`, each symbol should appear at most once in the list, and the symbols should be ordered according to the following comparison procedure:

   ```
   (define (symbol<? a b) (string<? (symbol->string a) (symbol->string b)))
   ```

   - Hint 1: First, write a function that generates an unordered list of symbols with duplicates, and then write separate functions to re-order the list and then remove duplicates.
   - Hint 2: The free variables of a **`with`** form include all of the free variable of the named expression, plus all of the free variables in the body expression except for the name bound by **`with`**.
   - Hint 3: The PLAI language includes several helpful list-processing functions: `filter`, `sort`, `member, ...`
   - http://docs.racket-lang.org/reference/pairs.html

2. **`; binding-ids : WAE -> list-of-sym`**
   Implement the function `binding-ids`, which is like `free-ids`, but the result list contains a symbol for each binding identifier in the given `WAE` (whether or not the binding identifier is ever referenced by a bound identifier). The result list of symbols must be sorted and have no duplicates.

3. **`; bound-ids : WAE -> list-of-sym`**
   Implement the function `bound-ids`, which is like `free-ids`, but the result list contains a symbol for each bound identifier in the given `WAE`. The result list of symbols must be sorted and have no duplicates.

Here are some tests:

```
;; free-ids
(test (free-ids (with 'x (num 3) (add (id 'x) (sub (num 3) (id 'x))))) '())
(test (free-ids (with 'x (num 3) (sub (id 'a) (add (num 4) (id 'x))))) '(a))
(test (free-ids (with 'x (num 3) (sub (id 'b) (sub (id 'a) (id 'x)))))
                 '(a b))

(test (free-ids (with 'x (num 3) (sub (id 'a)
                                      (sub (id 'b) (add (id 'x) (id 'b))))))
      '(a b))
```

```
(test (free-ids (with 'x (num 3)
                      (sub (id 'y)
                           (with 'y (num 7)
                                 (add (id 'x) (sub (id 'b) (id 'a)))))))
      '(a b y))

(test (free-ids (with 'x (id 't)
                      (sub (id 'x)
                           (with 'y (id 'y)
                                 (add (id 'x) (sub (id 'b) (id 'a)))))))
      '(a b t y))

(test (free-ids (with 'x (with 'y (num 3) (sub (id 'x) (id 'y)))
                      (add (id 'x) (id 'y))))
      '(x y))

(test (free-ids (add (with 'x (num 10)
                           (with 'x (num 3) (sub (id 'y)
                                 (with 'y (num 7) (add (id 'x)
                                                  (sub (id 'c) (id 'b)))))))
                     (with 'a (id 'a) (id 'a))))
      '(a b c y))

(test (free-ids (add (with 'x (num 10)
                           (with 'x (num 3)
                                 (sub (id 'y)
                                      (with 'y (num 7)
                                            (add (id 'x)
                                                 (sub (id 'c) (id 'b)))))))
                     (with 'a (id 'd) (id 'a))))
      '(b c d y))

(test (free-ids (add (with 'x (num 10)
                           (with 'x (num 3)
                                 (sub (id 'y)
                                      (with 'y (num 7)
                                            (add (id 'x)
                                                 (sub (id 'c) (id 'b)))))))
                     (with 'a (id 'd) (id 'z))))
      '(b c d y z))

;; binding-ids
(test (binding-ids (add (num 3) (sub (id 'x) (id 'y)))) '())

(test (binding-ids (with 'y (num 3) (with 'x (id 'x) (id 'y)))) '(x y))

(test (binding-ids (with 'y (num 3)
                         (with 'y (id 'x) (add (id 'x) (id 'y)))))
      '(y))

(test (binding-ids (with 'y (num 3)
                         (with 'y (with 'x (add (num 3) (id 'y))
                                        (sub (id 'x) (id 'y)))
                               (add (id 'x) (id 'y)))))
      '(x y))
```

```
(test (binding-ids (with 'z (num 3)
                         (with 'w (with 'z (add (num 3) (id 'y))
                                        (sub (id 'x) (id 'y)))
                               (with 'w (id 'y) (add (num 7) (id 'w))))))
      '(w z))

;; bound-ids

(test (bound-ids (with 'x (num 3) (add (id 'y) (num 3)))) '())

(test (bound-ids (with 'x (num 3) (add (id 'x) (sub (id 'x) (id 'y))))) '(x))

(test (bound-ids (with 'x (num 3) (add (id 'x) (with 'y (num 7) (sub (id 'x)
(id 'y)))))) '(x y))

(test (bound-ids (with 'x (num 3) (with 'y (id 'x) (sub (num 3) (id 'y)))))
'(x y))

(test (bound-ids (with 'x (num 3) (add (id 'y) (with 'y (id 'x) (sub (num 3)
(num 7)))))) '(x))

(test (bound-ids (with 'x (id 'x) (add (id 'y) (with 'y (id 'y) (sub (num 3)
(with 'z (num 7) (sub (id 'z) (id 'x)))))))) '(x z))

(test (bound-ids (with 'x (with 'y (num 3) (add (id 'x) (id 'y))) (add (id
'y) (with 'y (id 'y) (sub (num 3) (num 7)))))) '(y))

(test (bound-ids (with 'x (id 'a) (with 'y (id 'b) (with 'z (id 'c) (add (id
'd) (sub (id 'x) (add (id 'y) (id 'z)))))))) '(x y z))

(test (bound-ids (add (with 'x (num 10) (with 'x (num 3) (sub (id 'y) (with
'y (num 7) (add (id 'x) (sub (id 'c) (id 'b))))))) (with 'a (id 'd) (id
'a)))) '(a x))

(test (bound-ids (add (with 'x (num 10) (with 'x (num 3) (sub (id 'y) (with
'y (num 7) (add (id 'x) (sub (id 'c) (id 'b))))))) (with 'a (id 'd) (id
'z)))) '(x))
```