```
Test Name: pretty-tests
Definitions:
         (define EXAMPLE
           (make-add
            (list (make-mul (list 1 2) true) (make-add (list 3 4)
false))
            true))
         (define get-width
           (lambda (l) (foldr max 0 (map (lambda (i) (string-length i))
1))))
         (define EXPR
           (make-add
            (list
             (make-mul (build-list 20 add1) false)
             (make-mul (build-list 20 add1) false))
            false))
         (define EXPR-ONE-LINE-LIST
           (list
            "(+ (* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)
(* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20))"))
         (define EXPR-TWO-LINE-LIST
           (list
            "(+ (* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)"
               (* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20))"))
         (define EXPR-MAX-LINE-LIST
           (list
            "(+ (* 1"
                   2"
            п
                   3"
            п
                   4"
            п
                   5"
            п
                   6"
                   7"
                   811
                   gu
                   10"
            п
                   11"
            п
                   12"
                   13"
                   14"
            п
                   15"
            п
                   16"
            п
                   17"
                   18"
                   19"
            п
                   20)"
```

```
11
                 (* 1"
            п
                    2"
            11
                    3"
                    4"
                    5"
            н
                    6"
                    7"
            п
                    8"
                    9"
                    10"
            п
                    11"
                    12"
            п
                    13"
                    14"
                    15"
            п
                    16"
            п
                    17"
            п
                    18"
            п
                    19"
                    20))"))
         (define EXPR-21-LINE-LIST
           (list
            "(+ (* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)"
                 (* 1"
            п
                    2"
                    3"
                    4"
            п
                    5"
                    6"
            п
                    7"
                    8"
                    9"
            п
                    10"
                    11"
            п
                    12"
                    13"
                    14"
                    15"
                    16"
            11
                    17"
            11
                    18"
            п
                    19"
                    20))"))
Test Case:
  (test-equal?
   "inline infix flat add"
   (expr->strings (make-add (list 10 20 30 40) true) 50)
   (list "(10 + 20 + 30 + 40)"))
Test Result: Success
```

```
Test Case:
  (test-equal?
   "inline infix nested add"
   (expr->strings
    (make-add
     (list
      (make-add (list 10 20) true)
      (make-add (list 30 40) true)
      (make-add (list 50 60 70) true))
     true)
    50)
   (list "((10 + 20) + (30 + 40) + (50 + 60 + 70))"))
Test Result: Success
Test Case:
  (test-equal?
   "inline infix flat mul"
   (expr->strings (make-mul (list 10 20 30) true) 50)
   (list "(10 * 20 * 30)"))
Test Result: Success
Test Case:
  (test-equal?
   "inline infix nested mul"
   (expr->strings
    (make-mul
     (list (make-mul (list 10 20) true) (make-mul (list 30 40) true))
     true)
    50)
   (list "((10 * 20) * (30 * 40))"))
Test Result: Success
Test Case:
  (test-equal?
   "inline prefix flat add"
   (expr->strings (make-add (list 10 20 30) false) 50)
   (list "(+ 10 20 30)"))
Test Result: Success
Test Case:
  (test-equal?
   "inline prefix nested add"
   (expr->strings
    (make-add
     (list (make-add (list 10 20) false) (make-add (list 30 40)
false))
     false)
    50)
   (list "(+ (+ 10 20) (+ 30 40))"))
```

```
Test Result: Success
Test Case:
  (test-equal?
   "inline prefix flat mul"
   (expr->strings (make-mul (list 10 20 30) false) 50)
   (list "(* 10 20 30)"))
Test Result: Success
Test Case:
  (test-equal?
   "inline prefix nested mul"
   (expr->strings
    (make-mul
     (list (make-mul (list 10 20) false) (make-mul (list 30 40)
false))
     false)
    50)
   (list "(* (* 10 20) (* 30 40))"))
Test Result: Success
Test Case:
  (test-equal?
   "inline mixed"
   (expr->strings
    (make-mul
     (list
      (make-add (list 10 20 50) true)
      (make-mul (list 30 40 60) false)
      70
      80)
     false)
    50)
   (list "(* (10 + 20 + 50) (* 30 40 60) 70 80)"))
Test Result: Success
Test Case:
  (test-equal?
   "multi-line infix flat add"
   (expr->strings (make-add (list 10 20 30) true) 6)
   (list "(10" " +" " 20" " +" " 30)"))
Test Result: Success
Test Case:
  (test-equal?
   "multi-line infix nested add"
   (expr->strings
    (make-add
     (list (make-add (list 10 20) true) (make-add (list 30 40) true))
     true)
```

```
10)
   (list "((10 + 20)" " +" " (30" " +" " 40))"))
Test Result: Success
Test Case:
  (test-exn
   "doesn't fit"
   exn:fail?
   (λ ()
     (expr->strings
      (make-add
       (list (make-add (list 10 20) true) (make-add (list 30 40)
true))
       true)
      5)))
Test Result: Success
Test Case:
  (test-equal?
   "multi-line infix flat mul"
   (expr->strings (make-mul (list 10 20 30) true) 6)
   (list "(10" " *" " 20" " *" " 30)"))
Test Result: Success
Test Case:
  (test-equal?
   "multi-line infix nested mul"
   (expr->strings
    (make-mul
     (list (make-mul (list 10 20) true) (make-mul (list 30 40) true))
     true)
    10)
   (list "((10 * 20)" " *" " (30" " *" " 40))"))
Test Result: Success
Test Case:
  (test-exn
   "doesn't fit"
   exn:fail?
   (\lambda () (expr->strings (make-add (list 10 20 30) false) 5)))
Test Result: Success
Test Case:
  (test-equal?
   "multi-line prefix flat add"
   (expr->strings (make-add (list 10 20 30) false) 6)
   (list "(+ 10" "
                     20" "
                              30)"))
Test Result: Success
Test Case:
```

```
(test-equal?
   "multi-line prefix nested add"
   (expr->strings
    (make-add
     (list (make-add (list 10 20) false) (make-add (list 30 40)
false))
     false)
    10)
                            20)" "
                                     (+ 30" "
   (list "(+ (+ 10" "
                                                    40))"))
Test Result: Success
Test Case:
  (test-equal?
   "multi-line prefix flat mul"
   (expr->strings (make-mul (list 10 20 30) false) 6)
   (list "(* 10<sup>"</sup> "
                      20" "
                              30)"))
Test Result: Failure
actual : ((+ 10
                   20
                          30))
expected: ((* 10
                     20
                            30))
expression: (check-equal? (expr->strings (make-mul (list 10 20 30)
false) 6) (list (* 10
                          20
                                30)))
params : (((+ 10
                    20
                           30)) ((* 10
                                          20
                                                 30)))
Test Case:
  (test-equal?
   "multi-line prefix nested mul"
   (expr->strings
    (make-mul
     (list (make-mul (list 10 20) false) (make-mul (list 30 40)
false))
     false)
    10)
   (list "(* (* 10" "
                            20)" "
                                     (* 30" "
                                                    40))"))
Test Result: Failure
actual : ((+ (+ 10
                          20)
                                 (+30)
                                             40)))
expected : ((* (* 10
                            20)
                                   (* 30
                                                40)))
expression: (check-equal? (expr->strings (make-mul (list (make-mul
(list 10 20) false) (make-mul (list 30 40) false)) false) 10) (list (*
                                40))))
                   (* 30
            20)
params : (((+ (+ 10))
                           20)
                                  (+30)
                                              40))) ((* (* 10
                                                                      20)
(* 30
            40))))
Test Case:
  (test-equal?
   "multi-line mixed"
   (expr->strings
    (make-add
     (list (make-mul (list 10 20) false) (make-mul (list 30 40) true))
     false)
    10)
```

```
20)" " (30" "
   (list "(+ (* 10" "
                                               *" "
                                                       40))"))
Test Result: Failure
actual : ((+ (+ 10
                         20)
                                 (30
                                         *
                                               40)))
expected : ((+ (* 10
                           20)
                                   (30
                                                 40)))
expression: (check-equal? (expr->strings (make-add (list (make-mul
(list 10 20) false) (make-mul (list 30 40) true)) false) 10) (list (+
(* 10
            20)
                   (30
                                  40))))
                           *
params : (((+ (+ 10
                          20)
                                  (30
                                                40))) ((+ (* 10
                                         *
       (30
              *
                     40))))
Test Case:
  (test-equal?
   "unstacked"
   (expr->strings EXAMPLE 100)
   (list "((1 * 2) + (+ 3 4))"))
Test Result: Success
Test Case:
  (test-equal?
   "one level stacking"
   (expr->strings EXAMPLE 9)
   (list "((1 * 2)" " +" " (+ 3 4))"))
Test Result: Success
Test Case:
  (test-equal?
   "two level stacking (1 subexpression)"
   (expr->strings EXAMPLE 8)
   (list "((1 * 2)" " +" " (+ 3" "
                                       4))"))
Test Result: Success
Test Case:
  (test-equal?
   "two level stacking (2 subexpressions)"
   (expr->strings EXAMPLE 7)
   (list "((1" *" " 2)" " +" " (+ 3" "
                                               4))"))
Test Result: Success
Test Case:
  (test-exn "doesn't fit" exn:fail? (\lambda () (expr->strings EXAMPLE 6)))
Test Result: Success
Test Case:
  (test-equal?
   "Enough room to render on one line"
   (expr->strings EXPR (get-width EXPR-ONE-LINE-LIST))
   EXPR-ONE-LINE-LIST)
Test Result: Success
Test Case:
```

```
(test-equal?
   "bounds that are only slightly too narrow to render it on 1 line
should have 2 lines."
   (expr->strings EXPR (- (get-width EXPR-ONE-LINE-LIST) 1))
   EXPR-TWO-LINE-LIST)
Test Result: Success
Test Case:
  (test-equal?
   "bounds that are only slightly too narrow to render it on 2 lines
should have 21 lines."
   (expr->strings EXPR (- (get-width EXPR-TWO-LINE-LIST) 1))
   EXPR-21-LINE-LIST)
Test Result: Failure
actual : ((+ (* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)
(+ 1)
           2
                    3
10
         11
                   12
                             13
                                      14
                                                15
                                                          16
                                                                   17
18
         19
                   20))
expected: ((+ (* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)
(* 1
           2
                    3
                             4
                                     5
                                              6
                                                      7
                                                               8
                                                                       9
                   12
                             13
10
         11
                                      14
                                                15
                                                                   17
                                                          16
18
         19
                   20)))
expression: (check-equal? (expr->strings EXPR (- (get-width EXPR-TWO-
LINE-LIST) 1)) EXPR-21-LINE-LIST)
params : (((+ (* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)
                    3
(+ 1)
           2
                             4
                                     5
                                              6
                                                                       9
                   12
                             13
                                                         16
10
         11
                                      14
                                                15
                                                                   17
                   20)) ((+ (* 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
18
         19
17 18 19 20)
                             2
                                     3
                                              4
                                                      5
                                                               6
                 (* 1
        9
                 10
                                                       14
8
                           11
                                    12
                                              13
                                                                 15
                                      20))))
16
         17
                   18
                             19
Test Case:
  (test-equal?
   "bounds that are only slightly too narrow to render it on 21 lines
should have 40 lines."
   (expr->strings EXPR (- (get-width EXPR-21-LINE-LIST) 1))
   EXPR-MAX-LINE-LIST)
Test Result: Failure
actual : ((+ (+ 1
                                                   5
                                                                    7
                                           4
                                                            6
                         2
        9
                          11
                                                                 15
8
                 10
                                    12
                                              13
                                                       14
16
         17
                   18
                             19
                                      20)
                                              (+ 1)
                                                         2
                                                                  3
                         7
                                  8
                                           9
4
        5
                 6
                                                   10
                                                             11
                                                                      12
                                                          19
13
         14
                   15
                            16
                                      17
                                                18
                                                                   20))
expected: ((+ (* 1
                           2
                                    3
                                                     5
                                                              6
                                                                      7
                                             4
                 10
                          11
                                    12
                                                       14
                                                                 15
8
        9
                                              13
16
         17
                   18
                             19
                                      20)
                                              (* 1
                                                          2
                                                                  3
                         7
                                  8
                                           9
4
        5
                 6
                                                   10
                                                             11
                                                                      12
13
                   15
                                                18
                                                          19
                             16
                                      17
                                                                   20)))
expression: (check-equal? (expr->strings EXPR (- (get-width EXPR-21-
```

```
LINE-LIST) 1)) EXPR-MAX-LINE-LIST)
2
                                  3
                                                  5
                                                          6
                                                                   7
                                          4
                                   12
        9
                          11
                10
                                                     14
                                                               15
8
                                            13
                                                               3
         17
                                            (+1)
16
                  18
                            19
                                     20)
                                                       2
        5
                6
                        7
                                 8
4
                                         9
                                                 10
                                                           11
                                                                    12
13
                  15
                            16
                                              18
                                                       19
                                                                 20))
         14
                                     17
((+(*1
               2
                       3
                                4
                                        5
                                                6
                                                        7
                                                                8
9
        10
                 11
                          12
                                    13
                                             14
                                                      15
                                                                16
                                                                       5
17
         18
                  19
                           20)
                                   (* 1
                                              2
                                                      3
                                                               4
                8
                        9
                                                             13
6
        7
                                 10
                                          11
                                                   12
                                                                      14
                                              20))))
15
                                     19
         16
                  17
                            18
Test Case:
  (test-exn
   "EXPR doesnt fit"
   exn:fail?
   (lambda () (expr->strings EXPR (- (get-width EXPR-MAX-LINE-LIST)
1))))
Test Result: Failure
message : No exception raised
expression: (check-exn exn:fail? (lambda () (expr->strings EXPR (-
(get-width EXPR-MAX-LINE-LIST) 1))))
params : (#cedure:exn:fail?> #rocedure:temp132>)
Results for Suite pretty-tests:
  Test Successes: 24
  Test Failures: 6
  Test Errors: 0
Raw Score: 24/30
Normalized Score: 8/10
Overall Results:
  Test Successes: 24
  Test Failures: 6
  Test Errors: 0
Raw Score: 24/30
```

Normalized Score: 8/10