

```
1: #!/afs/cats.ucsc.edu/courses/cmcs112-wm/usr/racket/bin/mzscheme -qr
2: ;; $Id: sbi.scm,v 1.2 2015-09-23 17:11:09-07 - - $
3: ;;
4: ;; NAME
5: ;;     sbi.scm - silly basic interpreter
6: ;;
7: ;; SYNOPSIS
8: ;;     sbi.scm filename.sbir
9: ;;
10: ;; DESCRIPTION
11: ;;     The file mentioned in argv[1] is read and assumed to be an
12: ;;     SBIR program, which is then executed. Currently it is only
13: ;;     printed.
14: ;;
15:
16: (define *stderr* (current-error-port))
17:
18: (define *run-file*
19:   (let-values
20:     (((dirpath basepath root?)
21:       (split-path (find-system-path 'run-file))))
22:     (path->string basepath)))
23: )
24:
25: (define (die list)
26:   (for-each (lambda (item) (display item *stderr*)) list)
27:   (newline *stderr*)
28:   (exit 1))
29: )
30:
31: (define (usage-exit)
32:   (die `("Usage: " ,*run-file* " filename")))
33: )
34:
35: (define (readlist-from-inputfile filename)
36:   (let ((inputfile (open-input-file filename)))
37:     (if (not (input-port? inputfile))
38:         (die `(*run-file* ": " ,filename ": open failed"))
39:         (let ((program (read inputfile)))
40:           (close-input-port inputfile)
41:           program))))
42:
43: (define (write-program-by-line filename program)
44:   (printf "====~n")
45:   (printf "~a: ~s~n" *run-file* filename)
46:   (printf "====~n")
47:   (printf "(~n")
48:   (map (lambda (line) (printf "~s~n" line)) program)
49:   (printf ")~n"))
50:
51: (define (main arglist)
52:   (if (or (null? arglist) (not (null? (cdr arglist))))
53:       (usage-exit)
54:       (let* ((sbprogfile (car arglist))
55:              (program (readlist-from-inputfile sbprogfile)))
56:         (write-program-by-line sbprogfile program)))
57:
58: (main (vector->list (current-command-line-arguments)))
```

59:

```
1: ;;File: 00-hello-world.sb
2: ;; 1: # $Id: 00-hello-world.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # Classic Hello World program.
5: ;; 4: #
6: ;; 5: print "Hello, World!"
7: (
8: ( 1 )
9: ( 2 )
10: ( 3 )
11: ( 4 )
12: ( 5 (print "Hello, World!"))
13: )
```

```
1: ;;File: 01-1to10.sb
2: ;; 1: # $Id: 01-1to10.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # Print the numbers 1 to 10, one number per line.
5: ;; 4: #
6: ;; 5: print 1
7: ;; 6: print 2
8: ;; 7: print 3
9: ;; 8: print 4
10: ;; 9: print 5
11: ;; 10: print 6
12: ;; 11: print 7
13: ;; 12: print 8
14: ;; 13: print 9
15: ;; 14: print 10
16: (
17: ( 1 )
18: ( 2 )
19: ( 3 )
20: ( 4 )
21: ( 5 (print 1))
22: ( 6 (print 2))
23: ( 7 (print 3))
24: ( 8 (print 4))
25: ( 9 (print 5))
26: ( 10 (print 6))
27: ( 11 (print 7))
28: ( 12 (print 8))
29: ( 13 (print 9))
30: ( 14 (print 10))
31: )
```

```
1: ;;File: 02-exprs.sb
2: ;; 1: # $Id: 02-exprs.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # some expressions using print
5: ;; 4:
6: ;; 5:      print "1+1      = ", 1+1
7: ;; 6:      print "2-2      = ", 2- 2
8: ;; 7:      print "3*3      = ", 3*3
9: ;; 8:
10: ;; 9:      print
11: ;; 10:
12: ;; 11:      print "4/9      = ", 4/9
13: ;; 12:      print "3*4+5*6 = ", 3*4+5*6
14: ;; 13:
15: (
16: ( 1 )
17: ( 2 )
18: ( 3 )
19: ( 4 )
20: ( 5 (print "1+1      = " (+ 1 1)))
21: ( 6 (print "2-2      = " (- 2 2)))
22: ( 7 (print "3*3      = " (* 3 3)))
23: ( 8 )
24: ( 9 (print))
25: ( 10 )
26: ( 11 (print "4/9      = " (/ 4 9)))
27: ( 12 (print "3*4+5*6 = " (+ (* 3 4) (* 5 6))))
28: ( 13 )
29: )
```

```
1: ;;File: 10-exprs.sb
2: ;; 1: # $Id: 10-exprs.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # All of the following should print something without error mess
ages.
5: ;; 4: # This program checks to see if expressions can be interpreted.
6: ;; 5: #
7: ;; 6:
8: ;; 7: let pi = 4 * atan(1)
9: ;; 8: let e = exp(1)
10: ;; 9:
11: ;; 10: print "1+1      = ", 1+1
12: ;; 11: print "2-2      = ", 2- 2
13: ;; 12: print "3*3      = ", 3*3
14: ;; 13: print "4/9      = ", 4/9
15: ;; 14: print "2^10     = ", 2^10
16: ;; 15: print "3*4+5*6 = ", 3*4+5*6
17: ;; 16: #
18: ;; 17: print "log(10)  = ", log(10)
19: ;; 18: print "sqrt(2)   = ", sqrt(2)
20: ;; 19: print "pi       = ", pi
21: ;; 20: print "e        = ", e
22: ;; 21: #
23: ;; 22: print "+1/+0    = ", +1/+0
24: ;; 23: print "-1/+0    = ", -1/+0
25: ;; 24: print "+1/-0    = ", +1/-0
26: ;; 25: print "-1/-0    = ", -1/-0
27: ;; 26: print "+0/+0    = ", +0/+0
28: ;; 27: print "-0/-0    = ", -0/-0
29: ;; 28: print "sqrt(-1) = ", sqrt(-1)
30: ;; 29: print "log(0)   = ", log(0)
31: ;; 30: #
32: ;; 31: print "6.02e23 = ", 6.02*10^23
33: ;; 32: print "(1+2)/7 = ", (1+2)/7
34: (
35: ( 1 )
36: ( 2 )
37: ( 3 )
38: ( 4 )
39: ( 5 )
40: ( 6 )
41: ( 7 (let pi (* 4 (atan 1))) )
42: ( 8 (let e (exp 1)) )
43: ( 9 )
44: ( 10 (print "1+1      = " (+ 1 1)))
45: ( 11 (print "2-2      = " (- 2 2)))
46: ( 12 (print "3*3      = " (* 3 3)))
47: ( 13 (print "4/9      = " (/ 4 9)))
48: ( 14 (print "2^10     = " (^ 2 10)))
49: ( 15 (print "3*4+5*6 = " (+ (* 3 4) (* 5 6))))
50: ( 16 )
51: ( 17 (print "log(10)  = " (log 10)))
52: ( 18 (print "sqrt(2)   = " (sqrt 2)))
53: ( 19 (print "pi       = " pi))
54: ( 20 (print "e        = " e))
55: ( 21 )
56: ( 22 (print "+1/+0    = " (/ (+ 1) (+ 0))))
57: ( 23 (print "-1/+0    = " (/ (- 1) (+ 0))))
```

```
58: ( 24      (print "+1/-0    = " (/ (+ 1) (- 0)))
59: ( 25      (print "-1/-0    = " (/ (- 1) (- 0)))
60: ( 26      (print "+0/+0    = " (/ (+ 0) (+ 0)))
61: ( 27      (print "-0/-0    = " (/ (- 0) (- 0)))
62: ( 28      (print "sqrt(-1) = " (sqrt (- 1)))
63: ( 29      (print "log(0)   = " (log 0))
64: ( 30      )
65: ( 31      (print "6.02e23  = " (* 6.02 (^ 10 23)))
66: ( 32      (print "(1+2)/7 = " (/ (+ 1 2) 7))
67: )
```

```
1: ;;File: 11-let.sb
2: ;; 1: # $Id: 11-let.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # test let
5: ;; 4: #
6: ;; 5:      let i = 1
7: ;; 6:      let j = i + 3
8: ;; 7:      let k = 8 * i + 9 / j
9: ;; 8:      print "i=", i
10: ;; 9:      print "j=", j
11: ;; 10:     print "k=", k
12: (
13: ( 1 )
14: ( 2 )
15: ( 3 )
16: ( 4 )
17: ( 5 (let i 1))
18: ( 6 (let j (+ i 3)))
19: ( 7 (let k (+ (* 8 i) (/ 9 j))))
20: ( 8 (print "i=" i))
21: ( 9 (print "j=" j))
22: ( 10 (print "k=" k))
23: )
```



```
1: ;;File: 12-let.sb
2: ;; 1: # $Id: 12-let.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2:
4: ;; 3: # Simple let without expressions.
5: ;; 4:
6: ;; 5:         let i = 6
7: ;; 6:         print i
8: ;; 7:         dim a(10)
9: ;; 8:         let a(i) = 9
10: ;; 9:         print a(i)
11: (
12: ( 1 )
13: ( 2 )
14: ( 3 )
15: ( 4 )
16: ( 5 (let i 6))
17: ( 6 (print i))
18: ( 7 (dim (a 10)))
19: ( 8 (let (a i) 9))
20: ( 9 (print (a i)))
21: )
```

```
1: ;;File: 20-goto.sb
2: ;; 1: # $Id: 20-goto.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3:      goto zero
5: ;; 4: four:  print "four"
6: ;; 5:      goto done
7: ;; 6: one:   print "one"
8: ;; 7:      goto two
9: ;; 8: three: print "three"
10: ;; 9:      goto four
11: ;; 10: two:  print "two"
12: ;; 11:      goto three
13: ;; 12: zero: print "zero"
14: ;; 13:      goto one
15: ;; 14: done:
16: (
17: ( 1 )
18: ( 2 )
19: ( 3 (goto zero))
20: ( 4 four (print "four"))
21: ( 5 (goto done))
22: ( 6 one (print "one"))
23: ( 7 (goto two))
24: ( 8 three (print "three"))
25: ( 9 (goto four))
26: ( 10 two (print "two"))
27: ( 11 (goto three))
28: ( 12 zero (print "zero"))
29: ( 13 (goto one))
30: ( 14 done )
31: )
```

```
1: ;;File: 21-let-if.sb
2: ;; 1: # $Id: 21-let-if.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3:      let i = 1
5: ;; 4: loop:  print i
6: ;; 5:      let i = i + 1
7: ;; 6:      if i <= 10 goto loop
8: (
9: ( 1      )
10: ( 2      )
11: ( 3      (let i 1))
12: ( 4 loop (print i))
13: ( 5      (let i (+ i 1)))
14: ( 6      (if (<= i 10) loop))
15: )
```

```
1: ;;File: 22-fibonacci.sb
2: ;; 1: # $Id: 22-fibonacci.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # Print out all Fibonacci numbers up to max.
5: ;; 4: #
6: ;; 5:      let max = 10^6
7: ;; 6:
8: ;; 7:      let fib0 = 0
9: ;; 8:      let fib1 = 1
10: ;; 9:      print "fib(", 0, ")=", fib0
11: ;; 10:      print "fib(", 1, ")=", fib1
12: ;; 11:      let i=1
13: ;; 12: loop:  let fib = fib0 + fib1
14: ;; 13:      let i=i+1
15: ;; 14:      print "fib(", i, ")=", fib
16: ;; 15:      let fib0 = fib1
17: ;; 16:      let fib1 = fib
18: ;; 17:      if fib <= max goto loop
19: (
20: ( 1 )
21: ( 2 )
22: ( 3 )
23: ( 4 )
24: ( 5 (let max (^ 10 6))
25: ( 6 )
26: ( 7 (let fib0 0)
27: ( 8 (let fib1 1)
28: ( 9 (print "fib(" 0 ")=" fib0))
29: ( 10 (print "fib(" 1 ")=" fib1))
30: ( 11 (let i 1)
31: ( 12 loop (let fib (+ fib0 fib1))
32: ( 13 (let i (+ i 1))
33: ( 14 (print "fib(" i ")=" fib))
34: ( 15 (let fib0 fib1)
35: ( 16 (let fib1 fib)
36: ( 17 (if (<= fib max) loop))
37: )
```

```
1: ;;File: 25-pi-e-fns.sb
2: ;; 1: # $Id: 25-pi-e-fns.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2:
4: ;; 3: print pi, e
5: ;; 4: let pi = 4 * atan(1)
6: ;; 5: let e = exp(1)
7: ;; 6: print "pi = ", pi
8: ;; 7: print "e = ", e
9: ;; 8:
10: ;; 9: print "sqrt ( pi ) = ", sqrt ( pi )
11: ;; 10: print "exp ( pi ) = ", exp ( pi )
12: ;; 11: print "log ( pi ) = ", log ( pi )
13: ;; 12: print "sin ( pi ) = ", sin ( pi )
14: ;; 13: print "cos ( pi ) = ", cos ( pi )
15: ;; 14: print "tan ( pi ) = ", tan ( pi )
16: ;; 15: print "acos ( pi ) = ", acos ( pi )
17: ;; 16: print "asin ( pi ) = ", asin ( pi )
18: ;; 17: print "atan ( pi ) = ", atan ( pi )
19: ;; 18: print "abs ( pi ) = ", abs ( pi )
20: ;; 19: print "ceil ( pi ) = ", ceil ( pi )
21: ;; 20: print "floor( pi ) = ", floor( pi )
22: ;; 21: print "round( pi ) = ", round( pi )
23: ;; 22:
24: (
25: ( 1 )
26: ( 2 )
27: ( 3 (print pi e))
28: ( 4 (let pi (* 4 (atan 1))))
29: ( 5 (let e (exp 1)))
30: ( 6 (print "pi = " pi))
31: ( 7 (print "e = " e))
32: ( 8 )
33: ( 9 (print "sqrt ( pi ) = " (sqrt pi)))
34: ( 10 (print "exp ( pi ) = " (exp pi)))
35: ( 11 (print "log ( pi ) = " (log pi)))
36: ( 12 (print "sin ( pi ) = " (sin pi)))
37: ( 13 (print "cos ( pi ) = " (cos pi)))
38: ( 14 (print "tan ( pi ) = " (tan pi)))
39: ( 15 (print "acos ( pi ) = " (acos pi)))
40: ( 16 (print "asin ( pi ) = " (asin pi)))
41: ( 17 (print "atan ( pi ) = " (atan pi)))
42: ( 18 (print "abs ( pi ) = " (abs pi)))
43: ( 19 (print "ceil ( pi ) = " (ceil pi)))
44: ( 20 (print "floor( pi ) = " (floor pi)))
45: ( 21 (print "round( pi ) = " (round pi)))
46: ( 22 )
47: )
```

[illegible]

```
1: ;;File: 31-big-o-.sb
2: ;; 1: # $Id: 31-big-o-.sbir,v 1.2 2014-10-29 14:27:31-07 - - $
3: ;; 2:
4: ;; 3: # Given the value of N1, is the following program guaranteed
5: ;; 4: # to terminate? If so, what is the big-O of time for terminatio
n?
6: ;; 5: # http://en.wikipedia.org/wiki/Collatz_conjecture
7: ;; 6:
8: ;; 7: # Big-O
9: ;; 8: # C: while (n>1)n=n&1?3*n+1:n/2;
10: ;; 9: # APL: L:->Lx1<N<-((|_N/2),3xN+1)[1=2|N]
11: ;; 10:
12: ;; 11: input N1
13: ;; 12: let i = 0
14: ;; 13: let n = N1
15: ;; 14: while: if n <= 1 goto done
16: ;; 15: let i = i + 1
17: ;; 16: let f = floor( n / 2 )
18: ;; 17: if n <> f * 2 goto odd
19: ;; 18: let n = f
20: ;; 19: goto while
21: ;; 20: odd: let n = n * 3 + 1
22: ;; 21: goto while
23: ;; 22: done: print N1, " loops ", i, " times."
24: (
25: ( 1 )
26: ( 2 )
27: ( 3 )
28: ( 4 )
29: ( 5 )
30: ( 6 )
31: ( 7 )
32: ( 8 )
33: ( 9 )
34: ( 10 )
35: ( 11 (input N1))
36: ( 12 (let i 0))
37: ( 13 (let n N1))
38: ( 14 while (if (<= n 1) done))
39: ( 15 (let i (+ i 1)))
40: ( 16 (let f (floor (/ n 2))))
41: ( 17 (if (<> n (* f 2)) odd))
42: ( 18 (let n f))
43: ( 19 (goto while))
44: ( 20 odd (let n (+ (* n 3) 1)))
45: ( 21 (goto while))
46: ( 22 done (print N1 " loops " i " times.))
47: )
```

```
1: ;;File: 32-factorial.sb
2: ;; 1: # $Id: 32-factorial.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # Factorial.
5: ;; 4: #
6: ;; 5: read: print "Factorial of:"
7: ;; 6: input x
8: ;; 7: # check the variable eof for a valid value or not.
9: ;; 8: if inputcount = -1 goto stop
10: ;; 9: if inputcount = 0 goto error
11: ;; 10: if x < 0 goto error
12: ;; 11: goto letfac
13: ;; 12: error: print "Invalid input."
14: ;; 13: goto read
15: ;; 14:
16: ;; 15: #
17: ;; 16: #
18: ;; 17: #
19: ;; 18:
20: ;; 19: letfac: let factorial = 1
21: ;; 20: let itor = 2
22: ;; 21: loop: if itor > x goto prt
23: ;; 22: let factorial = factorial * itor
24: ;; 23: let itor = itor + 1
25: ;; 24: goto loop
26: ;; 25: prt: print "factorial(", x, ") = ", factorial
27: ;; 26: goto read
28: ;; 27:
29: ;; 28: #
30: ;; 29: # end of file.
31: ;; 30: #
32: ;; 31:
33: ;; 32: stop: print "Program stopping."
34: (
35: ( 1 )
36: ( 2 )
37: ( 3 )
38: ( 4 )
39: ( 5 read (print "Factorial of:"))
40: ( 6 (input x))
41: ( 7 )
42: ( 8 (if (= inputcount (- 1)) stop))
43: ( 9 (if (= inputcount 0) error))
44: ( 10 (if (< x 0) error))
45: ( 11 (goto letfac))
46: ( 12 error (print "Invalid input.))
47: ( 13 (goto read))
48: ( 14 )
49: ( 15 )
50: ( 16 )
51: ( 17 )
52: ( 18 )
53: ( 19 letfac (let factorial 1))
54: ( 20 (let itor 2))
55: ( 21 loop (if (> itor x) prt))
56: ( 22 (let factorial (* factorial itor)))
57: ( 23 (let itor (+ itor 1)))
58: ( 24 (goto loop))
```



```
59: ( 25 prt      (print "factorial(" x ") = " factorial))
60: ( 26          (goto read))
61: ( 27          )
62: ( 28          )
63: ( 29          )
64: ( 30          )
65: ( 31          )
66: ( 32 stop    (print "Program stopping."))
67: )
```

```
1: ;;File: 33-quadratic.sb
2: ;; 1: # $Id: 33-quadratic.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # Quadratic equation solver
5: ;; 4: #
6: ;; 5:
7: ;; 6:      print "Quadratic Equation solver."
8: ;; 7: loop:  print "Input a, b, c"
9: ;; 8:      input a, b, c
10: ;; 9:      if inputcount <> 3 goto stop
11: ;; 10:      let q = sqrt( b ^ 2 - 4 * a * c )
12: ;; 11:      print "Equation: ", a, " * x ^ 2 +", b, " * x +", c
13: ;; 12:      print "root1 = ", ( - b + q ) / ( 2 * a )
14: ;; 13:      print "root2 = ", ( - b - q ) / ( 2 * a )
15: ;; 14:      goto loop
16: ;; 15: stop:
17: (
18: ( 1 )
19: ( 2 )
20: ( 3 )
21: ( 4 )
22: ( 5 )
23: ( 6      (print "Quadratic Equation solver.))
24: ( 7 loop  (print "Input a, b, c"))
25: ( 8      (input a b c))
26: ( 9      (if (<> inputcount 3) stop))
27: ( 10      (let q (sqrt (- (^ b 2) (* (* 4 a) c))))
28: ( 11      (print "Equation: " a " * x ^ 2 +" b " * x +" c))
29: ( 12      (print "root1 = " (/ (+ (- b) q) (* 2 a))))
30: ( 13      (print "root2 = " (/ (- (- b) q) (* 2 a))))
31: ( 14      (goto loop))
32: ( 15 stop )
33: )
```

```
1: ;;File: 40-sort-array.sb
2: ;; 1: # $Id: 40-sort-array.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: # sort numbers
5: ;; 4: #
6: ;; 5: # Input is a sequence of numbers ending with end of file.
7: ;; 6: # User is assumed to have not more than 100 numbers.
8: ;; 7: #
9: ;; 8:      let size = 100
10: ;; 9:      dim a( size )
11: ;; 10:      let max = 0
12: ;; 11: read:  input x
13: ;; 12:      if inputcount = -1 goto eof
14: ;; 13:      if inputcount < 1 goto error
15: ;; 14:      let max = max + 1
16: ;; 15:      let a( max ) = x
17: ;; 16:      if max < size goto read
18: ;; 17: eof:
19: ;; 18:      print ""
20: ;; 19:      print "unsorted"
21: ;; 20:      let i = 1
22: ;; 21: prt1p:  print "a(", i, ")=", a( i )
23: ;; 22:      let i = i + 1
24: ;; 23:      if i <= max goto prt1p
25: ;; 24:      let i = max
26: ;; 25: outer:  let j = 1
27: ;; 26: inner:  if a(j) <= a(j + 1 ) goto noswap
28: ;; 27:      let t = a(j)
29: ;; 28:      let a(j) = a(j+1)
30: ;; 29:      let a(j+1)=t
31: ;; 30: noswap:
32: ;; 31:      let j = j + 1
33: ;; 32:      if j <= i - 1 goto inner
34: ;; 33:      let i = i - 1
35: ;; 34:      if i >= 2 goto outer
36: ;; 35:      print ""
37: ;; 36:      print "sorted"
38: ;; 37:      let i = 1
39: ;; 38: sort1p:  print "a(", i, ")=", a( i )
40: ;; 39:      let i = i + 1
41: ;; 40:      if i <= max goto sort1p
42: ;; 41:      goto stop
43: ;; 42: error:  print "Invalid input"
44: ;; 43: stop:
45: (
46: ( 1 )
47: ( 2 )
48: ( 3 )
49: ( 4 )
50: ( 5 )
51: ( 6 )
52: ( 7 )
53: ( 8 (let size 100))
54: ( 9 (dim (a size)))
55: ( 10 (let max 0))
56: ( 11 read (input x))
57: ( 12 (if (= inputcount (- 1)) eof))
58: ( 13 (if (< inputcount 1) error))
```

```
59: ( 14      (let max (+ max 1)))
60: ( 15      (let (a max) x))
61: ( 16      (if (< max size) read))
62: ( 17 eof   )
63: ( 18      (print ""))
64: ( 19      (print "unsorted"))
65: ( 20      (let i 1))
66: ( 21 prtlp (print "a(" i ")=" (a i)))
67: ( 22      (let i (+ i 1)))
68: ( 23      (if (<= i max) prtlp))
69: ( 24      (let i max))
70: ( 25 outer (let j 1))
71: ( 26 inner (if (<= (a j) (a (+ j 1))) noswap))
72: ( 27      (let t (a j)))
73: ( 28      (let (a j) (a (+ j 1))))
74: ( 29      (let (a (+ j 1)) t))
75: ( 30 noswap )
76: ( 31      (let j (+ j 1)))
77: ( 32      (if (<= j (- i 1)) inner))
78: ( 33      (let i (- i 1)))
79: ( 34      (if (>= i 2) outer))
80: ( 35      (print ""))
81: ( 36      (print "sorted"))
82: ( 37      (let i 1))
83: ( 38 sortlp (print "a(" i ")=" (a i)))
84: ( 39      (let i (+ i 1)))
85: ( 40      (if (<= i max) sortlp))
86: ( 41      (goto stop))
87: ( 42 error  (print "Invalid input"))
88: ( 43 stop   )
89: )
```

```
1: ;;File: 41-eratosthenes.sb
2: ;; 1: # $Id: 41-eratosthenes.sbir,v 1.1 2012-01-19 17:13:35-08 - - $
3: ;; 2: #
4: ;; 3: let n = 100
5: ;; 4: dim sieve(n)
6: ;; 5:
7: ;; 6: # Assume all numbers in the sieve are prime
8: ;; 7:
9: ;; 8: let i = 2
10: ;; 9: init: let sieve(i) = 1
11: ;; 10: let i = i + 1
12: ;; 11: if i < n goto init
13: ;; 12:
14: ;; 13: # Find primes and punch out their multiples.
15: ;; 14:
16: ;; 15: let prime = 2
17: ;; 16: primes: if sieve(prime) = 0 goto next
18: ;; 17: print prime
19: ;; 18: let i = prime * 2
20: ;; 19: goto punch
21: ;; 20: loop: let sieve(i) = 0
22: ;; 21: let i = i + prime
23: ;; 22: punch: if i <= n goto loop
24: ;; 23:
25: ;; 24: next: let prime = prime + 1
26: ;; 25: if prime <= n goto primes
27: (
28: ( 1 )
29: ( 2 )
30: ( 3 (let n 100))
31: ( 4 (dim (sieve n)))
32: ( 5 )
33: ( 6 )
34: ( 7 )
35: ( 8 (let i 2))
36: ( 9 init (let (sieve i) 1))
37: ( 10 (let i (+ i 1)))
38: ( 11 (if (< i n) init))
39: ( 12 )
40: ( 13 )
41: ( 14 )
42: ( 15 (let prime 2))
43: ( 16 primes (if (= (sieve prime) 0) next))
44: ( 17 (print prime))
45: ( 18 (let i (* prime 2)))
46: ( 19 (goto punch))
47: ( 20 loop (let (sieve i) 0))
48: ( 21 (let i (+ i prime)))
49: ( 22 punch (if (<= i n) loop))
50: ( 23 )
51: ( 24 next (let prime (+ prime 1)))
52: ( 25 (if (<= prime n) primes))
53: )
```