```
1: # $Id: 00-hello-world.sb,v 1.1 2019-01-18 11:47:25-08 - - $
2: #
3: # Classic Hello World program.
4: #
5:         print "Hello, World!"
```

```
  1: # $Id: 01-1to10.sb,v 1.1 2019-01-18 11:47:25-08 - - $
  2: #
  3: # Print the numbers 1 to 10, one number per line.
  4: #
  5:         print  1
  6:         print  2
  7:         print  3
  8:         print  4
  9:         print  5
 10:         print  6
 11:         print  7
 12:         print  8
 13:         print  9
 14:         print 10
```

```
 1: # $Id: 02-exprs.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3: # some expressions using print
 4:
 5:         print "1+1      = ", 1+1
 6:         print "2-2      = ", 2- 2
 7:         print "3*3      = ", 3*3
 8:
 9:         print
10:
11:         print "4/9      = ", 4/9
12:         print "3*4+5*6  = ", 3*4+5*6
13:
```

```
1: # $Id: 03-input-print.sb,v 1.2 2019-01-18 11:56:49-08 - - $
2: #
3: # Simple input a variable then print
4:
5:          input x
6:          print "x = ", x
7:
```

```
 1: # $Id: 10-exprs.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3: # All of the following should print something without error messages.
 4: # This program checks to see if expressions can be interpreted.
 5: #
 6:
 7:         let pi = 4 * atan(1)
 8:         let e = exp(1)
 9:
10:         print "1+1      = ", 1+1
11:         print "2-2      = ", 2- 2
12:         print "3*3      = ", 3*3
13:         print "4/9      = ", 4/9
14:         print "2^10     = ", 2^10
15:         print "3*4+5*6  = ", 3*4+5*6
16:
17:         print "log(10)  = ", log(10)
18:         print "sqrt(2)  = ", sqrt(2)
19:         print "pi       = ", pi
20:         print "e        = ", e
21:
22:         print "+1/+0    = ", +1/+0
23:         print "-1/+0    = ", -1/+0
24:         print "+1/-0    = ", +1/-0
25:         print "-1/-0    = ", -1/-0
26:         print "+0/+0    = ", +0/+0
27:         print "-0/-0    = ", -0/-0
28:         print "sqrt(-1) = ", sqrt(-1)
29:         print "log(0)   = ", log(0)
30:
31:         print "6.02e23  = ", 6.02*10^23
32:         print "(1+2)/7  = ", (1+2)/7
```

```
 1: # $Id: 11-let.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3: # test let
 4: #
 5:          let i = 1
 6:          let j = i + 3
 7:          let k = 8 * i + 9 / j
 8:          print "i=", i
 9:          print "j=", j
10:          print "k=", k
```

```
1: # $Id: 12-let-dim.sb,v 1.1 2019-01-18 11:47:25-08 - - $
2:
3: # Simple let without expressions.
4:
5:         let i = 6
6:         print i
7:         dim a[10]
8:         let a[i] = 9
9:         print a[i]
```

```
 1: # $Id: 20-goto.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3:          goto zero
 4: four:    print "four"
 5:          goto done
 6: one:     print "one"
 7:          goto two
 8: three:   print "three"
 9:          goto four
10: two:     print "two"
11:          goto three
12: zero:    print "zero"
13:          goto one
14: done:
```

```
1: # $Id: 21-let-if.sb,v 1.1 2019-01-18 11:47:25-08 - - $
2: #
3:          let i = 1
4: loop:    print i
5:          let i = i + 1
6:          if i <= 10 goto loop
```

```
 1: # $Id: 22-fibonacci.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3: # Print out all Fibonacci numbers up to max.
 4: #
 5:          let max = 10^6
 6:
 7:          let fib0 = 0
 8:          let fib1 = 1
 9:          print "fib(", 0, ")=", fib0
10:          print "fib(", 1, ")=", fib1
11:          let i=1
12: loop:    let fib = fib0 + fib1
13:          let i=i+1
14:          print "fib(", i, ")=", fib
15:          let fib0 = fib1
16:          let fib1 = fib
17:          if fib <= max goto loop
```

```
 1: # $Id: 25-pi-e-fns.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2:
 3:         print pi, e
 4:         let pi = 4 * atan(1)
 5:         let e = exp(1)
 6:         print "pi = ", pi
 7:         print "e = ", e
 8:
 9:         print "sqrt ( pi ) = ", sqrt ( pi )
10:         print "exp  ( pi ) = ", exp  ( pi )
11:         print "log  ( pi ) = ", log  ( pi )
12:         print "sin  ( pi ) = ", sin  ( pi )
13:         print "cos  ( pi ) = ", cos  ( pi )
14:         print "tan  ( pi ) = ", tan  ( pi )
15:         print "acos ( pi ) = ", acos ( pi )
16:         print "asin ( pi ) = ", asin ( pi )
17:         print "atan ( pi ) = ", atan ( pi )
18:         print "abs  ( pi ) = ", abs  ( pi )
19:         print "ceil ( pi ) = ", ceil ( pi )
20:         print "floor( pi ) = ", floor( pi )
21:         print "round( pi ) = ", round( pi )
22:
```

```
 1: # $Id: 30-input-pi.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3:
 4:          let pi = 4 * atan(1)
 5:
 6:          print "RE: House Bill 246, Indiana State Legislature, 1897"
 7:          print "Enter 1 if it has passed and 0 if not."
 8:
 9:          input tennessee
10:          if tennessee = 0 goto prt
11:          let pi = 3
12: prt:     print "pi = ", pi
13:
14: # Print some random junk to exercise the interpreter.
15:
16:          print
17:          print
18:          print .3333333333333333333333333333333333333333
19:          let avogadro = 6.02e23
20:          print avogadro
```

```
 1: # $Id: 31-big-o-.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2:
 3: # Given the value of N1, is the following program guaranteed
 4: # to terminate?  If so, what is the big-O of time for termination?
 5: # http://en.wikipedia.org/wiki/Collatz_conjecture
 6:
 7: # Big-O
 8: # C:    while(n>1)n=n&1?3*n+1:n/2;
 9: # APL:  L:->Lxi1<N<-((|_N/2),3xN+1)[1=2|N]
10:
11:         input N1
12:         let i = 0
13:         let n = N1
14: while:  if n <= 1 goto done
15:         let i = i + 1
16:         let f = floor( n / 2 )
17:         if n <> f * 2 goto odd
18:         let n = f
19:         goto while
20: odd:    let n = n * 3 + 1
21:         goto while
22: done:   print N1, " loops ", i, " times."
```

```
 1: # $Id: 32-factorial.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3: # Factorial.
 4: #
 5: read:   print "Factorial of:"
 6:         input x
 7:         # check the variable eof for a valid value or not.
 8:         if eof = 1 goto stop
 9:         if x <> x goto error
10:         if x < 0 goto error
11:         goto letfac
12: error:  print "Invalid input."
13:         goto read
14:
15: #
16: #
17: #
18:
19: letfac: let factorial = 1
20:         let itor = 2
21: loop:   if itor > x goto prt
22:         let factorial = factorial * itor
23:         let itor = itor + 1
24:         goto loop
25: prt:    print "factorial(", x, ") = ", factorial
26:         goto read
27:
28: #
29: # end of file.
30: #
31:
32: stop:   print "Program stopping."
```

```
 1: # $Id: 33-quadratic.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3: # Quadratic equation solver
 4: #
 5:
 6:          print "Quadratic Equation solver."
 7: loop:    print "Input a, b, c"
 8:          input a, b, c
 9:          if eof = 1 goto stop
10:          let q = sqrt( b ^ 2 - 4 * a * c )
11:          print "Equation: ", a, " * x ^ 2 +", b, " * x +", c
12:          print "root1 = ", ( - b + q ) / ( 2 * a )
13:          print "root2 = ", ( - b - q ) / ( 2 * a )
14:          goto loop
15: stop:
```

```
 1: # $Id: 40-sort-array.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3: # sort numbers
 4: #
 5: # Input is a sequence of numbers ending with end of file.
 6: # User is assumed to have not more than 100 numbers.
 7: # Note that nan <> nan, other was x = x for all x that is not nan.
 8: #
 9:          let size = 100
10:          dim a[size]
11:          let max = 0
12: read:    input x
13:          if eof <> 0 goto eof
14:          if x <> x goto error
15:          let max = max + 1
16:          let a[max] = x
17:          if max < size goto read
18: eof:
19:          print ""
20:          print "unsorted"
21:          let i = 1
22: prtlp:   print "a[", i, "]=", a[i]
23:          let i = i + 1
24:          if i <= max goto prtlp
25:          let i = max
26: outer:   let j = 1
27: inner:   if a[j] <= a[j + 1] goto noswap
28:          let t = a[j]
29:          let a[j] = a[j+1]
30:          let a[j+1]=t
31: noswap:
32:          let j = j + 1
33:          if j <= i - 1 goto inner
34:          let i = i - 1
35:          if i >= 2 goto outer
36:          print ""
37:          print "sorted"
38:          let i = 1
39: sortlp:  print "a[", i, "]=", a[i]
40:          let i = i + 1
41:          if i <= max goto sortlp
42:          goto stop
43: error:   print "Invalid input"
44: stop:
```

```
 1: # $Id: 41-eratosthenes.sb,v 1.1 2019-01-18 11:47:25-08 - - $
 2: #
 3:          let n = 100
 4:          dim sieve[n]
 5:
 6: # Assume all numbers in the sieve are prime
 7:
 8:          let i = 2
 9: init:    let sieve[i] = 1
10:          let i = i + 1
11:          if i < n goto init
12:
13: # Find primes and punch out their multiples.
14:
15:          let prime = 2
16: primes:  if sieve[prime] = 0 goto next
17:          print prime
18:          let i = prime * 2
19:          goto punch
20: loop:    let sieve[i] = 0
21:          let i = i + prime
22: punch:   if i <= n goto loop
23:
24: next:    let prime = prime + 1
25:          if prime <= n goto primes
```

```
1: #!/bin/bash
2: # $Id: mk.build,v 1.1 2019-01-18 11:47:25-08 - - $
3: # Checksource and do the build.
4:
5: export PATH=$PATH:/afs/cats.ucsc.edu/courses/cmps112-wm/bin/
6: partnercheck 2>&1 | tee partnercheck.log
7: checksource Makefile README* *.ml* >checksource.log 2>&1
8: gmake >gmake.log 2>&1
```

```
 1: #!/bin/bash
 2: # $Id: mk.tests,v 1.1 2019-01-18 11:47:25-08 - - $
 3:
 4: for input in *.sb
 5: do
 6:    output=$(echo $input | sed 's/.sb$/.output/')
 7:    echo $0: starting ./sbinterp $input
 8:    ./sbinterp <$input >$output 2>&1
 9:    echo $0: finished ./sbinterp $input
10: done
```