bc Calculator

bc is "an arbitrary precision calculator language" with syntax similar to the C programming language. It is generally used by typing the command bc on a Unix command prompt and entering a mathematical expression, such as (1 + 3) * 2, whereupon 8 will be immediately outputted. bc can be executed as either a mathematical scripting language or as an interactive mathematical shell, similar to interactive mode (>>>) in Python.

All numbers and variable contents are fixed precision floating-point numbers whose precision (in decimal places) is determined by the global scale variable.

Output is generated by deliberately not assigning the result of a calculation to a variable...

```
 hives = 5
bees = 6
hives * bees
30
```

or by using a 'print' statement

```
 hives = 9
bees = 10
totalBees = hives * bees
print totalBees
90
```

Mathematical Operators

Exactly as C The following bc operators behave exactly like their C counterparts:

```
 +     -     *     /
+=    -=    *=    /=
++    --    <     >
==    !=    <=    >=
( )   [ ]   { }
```

Similar to C he modulus operators:

```
%     %=
```

... behave exactly like their C counterparts only when the global scale variable is set to 0, i.e. all calculations are integer-only. When scale is greater than 0 the modulus is calculated relative to the smallest positive value greater than zero.

Only Resembling C The operators:

```
^     ^=
```

... resemble the C bitwise exclusive-or operators, but are in fact the bc integer exponentiation operators.

Math Library Functions

If bc is invoked with the -l option, a math library is preloaded and the default scale is set to 20. The math functions will calculate their results to the scale set at the time of their call. The math library defines the following functions:

```
s (x)
```

The sine of x, x is in radians.

```
c (x)
```

The cosine of x, x is in radians.

```
a (x)
```

The arctangent of x, arctangent returns radians.

```
l (x)
```

The natural logarithm of x.

```
e (x)
```

The exponential function of raising e to the value x.

```
j (n,x)
```

The bessel function of integer order n of x.

In addition is the built-in square root function

```
sqrt(x)
```

Conditional Operators

The following logical operators...

```
&&      ||      !
```

are available for use in conditional statements (such as within an if statement).

Functions

Since the bc ^ operator only allows an integer power to its right, one of the first functions a bc user might write is a power function with a floating point exponent

```
 # A function to return the integer part of a number
define int(number) {
   auto oldscale
   oldscale = scale
   scale = 0
    number /= 1 /* round number down */
   scale = oldscale
   return number
}

# Use the fact that number^exponent == e^(exponent*log(number))
define power(number,exponent) {
   if (exponent == int(exponent)) {
      return number ^ exponent
   } else {
      return e( exponent * l(number) )
   }
}
```

User Input

A 'read' statement allows the interactive input of a number into a running calculation.

Variable & Array names and Control

```
 if(cond)...

while(cond)...

for(init;cond;update)...)
```

These are all the same as C

Example Program

Let us make a simple check book program together.

```
 scale=2
print "\nCheck book program\n!"
print "  Remember, deposits are negative transactions.\n"
print "  Exit by a 0 transaction.\n\n"

print "Initial balance? "; bal = read()
bal /= 1
print "\n"
while (1) {
  "current balance = "; bal
  "transaction? "; trans = read()
  if (trans == 0) break;
  bal -= trans
  bal /= 1
}
quit
```

Now let's do a recursive factorial function

```
 define f (x) {
  if (x <= 1) return (1);
  return (f(x-1) * x);
}
```