Further examples of input for testing extensions to your functional calculator

(suggested by the examination questions)

The other set of EGxx.FUN are still available if you want to reuse them.

You can (and probably should) devise some test cases of your own. Keep them simple.

Some of the ones below have deliberate errors, some will not run properly even if they compile properly.

After successfully using

CMAKE CalcPVM

you can test any little program called

Something.FUN

by issuing the command

Test Something

(no parameter required)

```
// ======================================================================

$C+ // generate .cod file for checking manually
// What could be more meaningless than this, but try it anyway!   EXAM00.FUN

        // function definition

        AssertInRange(x, l, h)  returns (x >= l) && (x <= h);

        // trial calls

        t = 400 > 300;
        read("Supply two numbers ", x, y);
        writeLine("Is x > y?", x > y);
        writeLine(AssertInRange(x, x, x));
        writeLine(AssertInRange(x, false, true) == true);

// ======================================================================

$C+ // generate .cod file for checking manually
// Very silly - what do you suppose happens and why?  EXAM01.FUN

        // function definition

        Silly(x)   returns 2 * Silly(x);

        // trial call

        write ( Silly(x) );

// ======================================================================

$C+ // generate .cod file for checking manually
// An example with errors   EXAM02.FUN

        // function definitions

        Sum(x, y, z) returns x + y;

        Product(x, y, z) returns  x * y * z;

        // trial call

        read("Supply three numbers ", x, y, z);
        writeLine("Their sum is ", Sun(x, y, z));
        writeLine("Their product is ", Product());
```

```
// ======================================================================

$C+ // generate .cod file for checking manually
// Example of recursive function - Factorial  EXAM03.FUN

        // function definition

        Factorial(n) returns
          if (n <= 0) returns 1;
            else returns n * Factorial(n - 1);

        // trial calls

        writeLine(" 0! =", Factorial(0));
        writeLine(" 1! =", Factorial(1));

        read ("Supply number ", n);
        writeLine(n, "! =", Factorial(n));

// ======================================================================

$C+ // generate .cod file for checking manually
// Example of recursive function - Factorial  EXAM03.FUN

        // function definition

        Factorial(n) returns
          if (n <= 0) returns 1;
            else returns n * Factorial(n - 1);

        // trial calls

        writeLine(" 0! =", Factorial(0));
        writeLine(" 1! =", Factorial(1));

        read ("Supply number ", n);
        writeLine(n, "! =", Factorial(n));

// ======================================================================

$C+ // generate .cod file for checking manually
// Example of recursive function - Fibonacci   EXAM05.FUN

        // function definition

        Fib(x) returns
          if (x == 0) returns 0;
          else if (x == 1) returns 1;
               else returns Fib(x - 1) + Fib(x - 2);

        // trial calls

        writeLine("fib(", 0, ") =",  Fib(0));
        writeLine("fib(", 1, ") =",  Fib(1));

        read ("Supply number ", x);
        writeLine("fib(", x, ") =",  Fib(x));

// ======================================================================

$C+ // generate .cod file for checking manually
// Example of recursive function - Fibonacci   EXAM05.FUN

        // function definition

        Fib(x) returns
          if (x == 0) 0;
          else if (x == 1) 1;
               else Fib(x - 1) + Fib(x - 2);

        // trial calls

        writeLine("fib(", 0, ") =",  Fib(0));
        writeLine("fib(", 1, ") =",  Fib(1));

        read ("Supply number ", x);
        writeLine("fib(", x, ") =",  Fib(x));
```

```
// ======================================================================

$C+ // generate .cod file for checking manually
// Another example with errors  EXAM06.FUN

        // function definitions

        AgeNow() returns 67;

        PlusOne(x) returns x + 1;

        // trial call

        writeLine("Pat will be", PlusOne(AgeNow), " next year");
        writeLine("Do you suppose he will live until he is",
                  PlusOne(PlusOne(PlusOne(AgeNow))), "?");

// ======================================================================

$C+ // generate .cod file for checking manually
// Another example with errors  EXAM07.FUN

        // function definitions

        Sqr(x) returns x * x;

        Sum(x, y) returns x + y;

        Sqr(y) returns y * y;

        // trial call

        read("Supply two numbers ", x, y);
        write("The sum of their squares is", Sum(Sqr(x), Sqr(y)));

// ======================================================================

$C+ // generate .cod file for checking manually
// Another example with errors  EXAM08.FUN

        // function definitions

        Sqr[x) returns x * x;

        Sqr(x) returns x * x;

        Sum(x, y, z) returns x + y + z;

        Product(x, y, z) returns x * y * z;

        // trial calls

        read ("Supply three numbers ", x, y, z);
        writeLine("x =", x, ", y =" , y , ", z =", z);
        writeLine("x squared =", Sqr(x),
                  ", Sum = " , Sum{x, y, z),
                  ", Product =", Product(x, y, z));

// ======================================================================

$C+ // generate .cod file for checking manually
// Another example with errors  EXAM09.FUN

        // function definitions

        Sum(x, y) returns + x + y;

        Product(x, y, z) returns x * y;

        // trial call

        read("Supply three numbers ", x, y, z);
        writeLine("Their sum is", Sum(x, y, z));
        writeLine("Their product is", Product(x, y));
```

```
// ======================================================================

$C+ // generate .cod file for checking manually
// Another example with errors   EXAM10.FUN

        // function definitions

        Sum(x, y, z) returns x + y;

        Product(x, y, z) returns  x * y * z;

        // trial call

        read("Supply three numbers ", x, y, z);
        writeLine("Their sum is", Sun(x, y, z));
        writeLine("Their product is", Product());

// ======================================================================

$C+ // generate .cod file for checking manually
// Another example with errors   EXAM11.FUN

        // function definitions

        Sqr[x) returns x * x;

        Sqr(x) returns x * x;

        Sum(x, y, z) returns x + y + z;

        Product(x, y, z) returns x * y * z;

        // trial call

        read ("Supply three numbers ", x, y, z);
        writeLine("x =", x, ", y =" , y , ", z =", z);
        writeLine("x squared =", Sqr(x),
                  ", Sum =" , Sum{x, y, z),
                  ", Product =", Product(x, y, z));
```