LispLisp i1

LispLisp is a simple Turing-complete programming language with a Lisp-like syntax, based on Chris Barker's universal combinator: $\iota := \lambda f.((fS)K)$, represented in LispLisp as the word "lisp."

Using This combinator, combinators S, K, and I can be represented as follows:

```
S := (lisp (lisp (lisp (lisp lisp))))
K := (lisp (lisp (lisp lisp)))
I := (lisp lisp)
```

Language Description

Each LispLisp program is made up of one or more blocks. Program blocks consist of LispLisp Code, entirely within either a pair of parenthesis or a pair of square brackets. Each block returns one value. Since LispLisp has no types, that value can be anything able to be represented in the Lambda calculus. Within parenthesis, that value is returned as-is (i.e. if the code within the block returns "48," the block returns 48.) Within square brackets, however, the value is returned as the corresponding ASCII character (i.e. if the code within the block returns "48," the block returns the character "0.") This means that blocks that would return numbers larger than 255 cannot be housed in square brackets. Blocks are interpreted sequentially as they appear in the program, from top to bottom. Comments, in typical Lisp fashion, can be constructed on a single line, beginning with `;` or on multiple lines with `#|` and `|#`.

Syntax Examples

The number 0 can represented as follows:

```
((lisp (lisp (lisp lisp)))
      (lisp lisp))
```

1 can be this:

```
(((lisp (lisp (lisp (lisp lisp))))
   (((lisp (lisp (lisp (lisp lisp))))
     ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp (lisp lisp))))))
    (((lisp (lisp (lisp (lisp lisp))))
```

```
      ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp lisp)))))
     (lisp lisp))))
  ((lisp (lisp (lisp lisp))) (lisp lisp)))
```

This code outputs the ASCII Bell Character (7):

(Notice the square brackets around the block of code)

```
[(((lisp (lisp (lisp (lisp lisp))))
   (((lisp (lisp (lisp (lisp lisp))))
     ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp (lisp lisp))))))
    (((lisp (lisp (lisp (lisp lisp))))
      ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp lisp)))))
     (lisp lisp))))
  (((lisp (lisp (lisp (lisp lisp))))
    (((lisp (lisp (lisp (lisp lisp))))
      ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp (lisp lisp))))))
     (((lisp (lisp (lisp (lisp lisp))))
       ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp lisp)))))
      (lisp lisp))))
   (((lisp (lisp (lisp (lisp lisp))))
     (((lisp (lisp (lisp (lisp lisp))))
       ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp (lisp lisp))))))
      (((lisp (lisp (lisp (lisp lisp))))
        ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp lisp)))))
       (lisp lisp))))
    (((lisp (lisp (lisp (lisp lisp))))
      (((lisp (lisp (lisp (lisp lisp))))
        ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp (lisp lisp))))))
       (((lisp (lisp (lisp (lisp lisp))))
         ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp lisp)))))
        (lisp lisp))))
     (((lisp (lisp (lisp (lisp lisp))))
       (((lisp (lisp (lisp (lisp lisp))))
         ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp (lisp lisp))))))
        (((lisp (lisp (lisp (lisp lisp))))
          ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp lisp)))))
         (lisp lisp))))
       (((lisp (lisp (lisp (lisp lisp))))
         (((lisp (lisp (lisp (lisp lisp))))
           ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp (lisp lisp))))))
```

```
  (((lisp (lisp (lisp (lisp lisp))))
    ((lisp (lisp (lisp lisp))) (lisp (lisp (lisp lisp)))))
   (lisp lisp))))
((lisp (lisp (lisp lisp))) (lisp lisp))))))))))]
```