Programming projects

Notes

✉ You don't have to do the projects in order! Do whatever seems fun at the moment.

✉ You don't have to do the projects at all! They're just here to inspire you. If you think of something cool, run with it—and please tell a helper about it, because we love cool things.

✉ When I say a computer is supposed to input or output a number, with no other details, I mean for the number to be represented as a tally, like this:

1	`o`
2	`oo`
3	`ooo`
4	`oooo`

Of course, any symbol will work just as well as `o`!

Basics

☞ Suppose your input is a string like this:

```
|---->--------|
```

Find rules that will make the `>` travel to the right until it hits the `|`, turn into a `<`, travel to the left until it hits the other `|`, turn into a `>`, and so on...

☞ Suppose your input is a string like this:

```
oooo|---->--------|
```

Find rules that will make the `>` bounce back and forth like in the previous project, but stop after it hits the left wall `oooo` times.

☞ Suppose your input is a string like this:

```
..T..T....T.......T.....T....TT......T...
```

Find rules that will make a `.` become a `<` if its nearest `T` is to the left, a `>` if its nearest `T` is to the right, and a `|` if it's equal distances between two `T`s.

☞ Write a computer that takes in a number n and a string of dashes, and turns every nth dash into a dot.

Multiplication

☞ IN Two numbers
OUT The product of the numbers

☞ IN A number
OUT The square of the number

☞ IN A number n
OUT The sum of the numbers from 1 to n

Numerals

☞ IN A number in binary
OUT The number (as a tally)

☞ IN A number (as a tally)
OUT The number in binary

☞ Write computers that convert back and forth between tallies and

- Maya numerals.
- Roman numerals.

☞ IN A number in binary
OUT The number in trinary

Division

☞ IN A number
OUT The number's remainder when divided by five

☞ IN Two numbers
OUT The first number's remainder when divided by the second

☞ The *Euclidean algorithm* is a method for finding the greatest common divisor of two numbers. It works like this.

Start with a pair of numbers. Subtract the smaller number from the larger number (if the numbers are equal, subtract either one from the other). Now you have a new pair of numbers. Keep subtracting the smaller from the larger until one of the numbers is gone. The remaining number is the greatest common divisor of the numbers you started with.

Write a computer that finds the greatest common divisor of two numbers.

☞ IN A number
OUT The empty string if the number is prime
A non-empty string otherwise

(The last **Basics** project might be helpful for this.)

Sequences

☞ Write a computer that will show you...

- all the numbers,
- all the Fibonacci numbers,

- all the prime numbers,

one by one.

☞ Write a computer that churns out the never-ending string

o_oo_ooo_oooo_ooooo_oooooo_ ...

(the ellipsis isn't part of the string).

☞ Take a strip of paper and fold it in half over and over, always folding in the same direction. Then, open each crease out into a 90° corner, keeping the natural direction of the crease. The shape you end up with is called a *dragon curve*.

Folding paper over and over gets difficult really quickly. It would be much easier to fold a dragon curve if you knew the sequence of left and right folds ahead of time—then you'd only have to fold one layer of paper.

Write a computer that will show you all the folding sequences for dragon curves.

Trees

IN A string of parentheses

☞ OUT The empty string if the parentheses are balanced
A non-empty string otherwise

IN A boolean expression—a tree with "and" or "or" at each node, and "true" or "false" at each leaf

☞ OUT The value of the expression