Mini-Logo

September 6, 2018

These practice exercises seek to help you get some acquaintance with lists and tuples.

1 Introduction

Encoding of Mini-Logo instructions:

Encoding	Instruction
0	Pen down
1	Pen up
2	Move North
3	Move East
4	Move South
5	Move West

A Mini-Logo program is just a list of instructions (integers). An example of a program that draws a square:

```
[0; 2; 3; 4; 5; 1]
```

This other one draws two concentric squares

```
[0;2; 2; 2; 2; 3; 3; 3; 3; 4; 4; 4; 4; 5; 5; 5; 5; 2; 2; 2; 2; 2; 2; 2; 2; 3; 3; 3; 3; 3; 3;
↪ 3; 3; 4; 4; 4; 4; 4; 4; 4; 4; 5; 5; 5; 5; 5; 5; 5; 5; 1]
```

1.1 Exercises

1. Implement a function

   ```
   coverage : int*int -> int list -> (int*int) list
   ```

   that given a starting coordinate and a program returns the list of coordinates that the program visits. You may introduce helper functions to make your code more readable.

2. Implement a function

   ```
   mirror_image : int list -> int list
   ```

that returns a program that draws the mirror image of the input program.

3. Implement a function

```
rotate_90 : int list -> int list
```

that given a program returns a new one which draws the same pictures except that they are rotated 90 degrees.

4. Implement a function

```
repeat : int -> 'a -> 'a list
```

such that `repeat n x` returns a list with `n` copies of `x`.

5. Implement a function

```
pantograph : int list -> int -> int list
```

such that `pantograph p n` returns a program that draws the same things as `p` only enlarged `n`-fold.

6. Implement a function

```
compress : int list -> (int*int) list
```

that compresses a program by replacing adjacent copies of the same instruction with a tuple `(m,n)` where `m` is the instruction and `n` is the number of consecutive times it should be executed.

7. Implement a function

```
decompress : (int*int) list -> int list
```

that decompresses a compressed program.