Mission Sidequest 21.1: Streams II

Start date: 31 October 2017
Due: 13 November 2017, 23:59

Readings:

- SICP, Section 3.5

You felt that the entertainment bot called `@realScottieBot` can be enhanced and you want to add more cool features to it. You consulted Grandmaster on this and he thus decided to offer you this ad-hoc training on stream.

This side quest consists of **four** tasks.

Task 1:

Recall Sidequest 20.1 Task 3, you have created a stream called `step_duration_stream`. Basically, it will return a stream that looks similar to `list (1, 2, 3, 1, 2, 3, ...)`. In this task, you are required to generalize the concept of `step_duration_stream`. Define a function called `make_step_stream` that takes a positive integer as the only parameter and returns a stream that looks similar to `list (1, 2, 3, ..., n, 1, 2, 3, ..., n, 1, ...)`. You can look at the sample output below.

```
var stream_123 = make_step_stream(3);
eval_stream(stream_123, 10);
// Output should be the same as list(1, 2, 3, 1, 2, 3, 1, 2, 3, 1)

var stream_12345 = make_step_stream(5);
eval_stream(stream_12345,10);
// Output should be the same as list(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)

var stream_1 = make_step_stream(1);
eval_stream(stream_1, 10);
// Output should be the same as list(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
```

Task 2:

Similarly, we want to generalize the concept of `oscillating_cookie_stream` mentioned in Sidequest 20.1 Task 3 as well. Define a function called `make_oscillating_stream` that has a behaviour exhibited in the sample execution below.

```
var osc_stream_123 = make_oscillating_stream(3);
eval_stream(osc_stream_123, 10);
// Output should be the same as list(1, 2, 3, 2, 1, 2, 3, 2, 1, 2)

var osc_stream_1 = make_oscillating_stream(1);
eval_stream(osc_stream_1, 10);
// Output should be the same as list(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)

var osc_stream_12345 = make_oscillating_stream(5);
eval_stream(osc_stream_12345,10);
// Output should be the same as list(1, 2, 3, 4, 5, 4, 3, 2, 1, 2)
```

Hint: `"oscillate"` means swinging backward and forward like a pendulum. Thus, similarly, your function should be able to *"switch"* between incrementing and decrementing.

Task 3:

While the function you defined above in Tasks 1 and 2 are functional, they are limited because we are unable to create arbitrary stepping patterns or oscillating patterns. Fortunately, we can get around this limitation easily.

Define the functions `make_flexible_step_stream` and `make_flexible_oscillating_stream` that exhibit the behaviour shown below.

```
var flex_123_step_stream = make_flexible_step_stream(list(1,2,3));
eval_stream(flex_123_step_stream, 10);
// Output should be the same as list(1, 2, 3, 1, 2, 3, 1, 2, 3, 1)

var flex_357_step_stream = make_flexible_step_stream(list(3,5,7));
eval_stream(flex_357_step_stream, 10);
// Output should be the same as list(3, 5, 7, 3, 5, 7, 3, 5, 7, 3)

var flex_1_step_stream = make_flexible_step_stream(list(1));
eval_stream(flex_1_step_stream, 10);
// Output should be the same as list(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)

var flex_123_osc_stream = make_flexible_oscillating_stream(list(1,2,3));
eval_stream(flex_123_osc_stream, 10);
// Output should be the same as list(1, 2, 3, 2, 1, 2, 3, 2, 1, 2)

var flex_3579_osc_stream = make_flexible_oscillating_stream(list(3,5,7,9));
eval_stream(flex_3579_osc_stream, 10);
// Output should be the same as list(3, 5, 7, 9, 7, 5, 3, 5, 7, 9)
```

```
var flex_12_osc_stream = make_flexible_oscillating_stream(list(1,2));
eval_stream(flex_12_osc_stream, 10);
// Output should be the same as list(1, 2, 1, 2, 1, 2, 1, 2, 1, 2)
```

Task 4:

In this last task, you are required to define a function called `interleave`. Do not confuse with the function `stream_zip` defined in Sidequest 20.1.

The function `interleave` produces a stream by interleaving two streams together. Note that the given streams may not necessarily be infinite and should be handled in the way shown below. See below for a sample execution.

```
// stream_constant(k) generates an infinite stream of k
function stream_constant(k) {
    return pair(k, function() { return stream_constant(k); });
}

// add_streams sums up two given infinite stream
function add_streams(s1, s2) {
    return pair( head(s1) + head(s2), function() {
        return add_streams( stream_tail(s1), stream_tail(s2));
    });
}

var odd_stream = pair(1, function(){
    return add_streams(stream_constant(2), odd_stream);
});
var even_stream = pair(2, function(){
    return add_streams(stream_constant(2), even_stream);
});

var integers = interleave(odd_stream, even_stream);
eval_stream(integers, 10);
// Output should be the same as list(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

var finite_test = interleave(list_to_stream(list("a","b","c")), stream_constant(1));
eval_stream(finite_test, 10);
// Output should be the same as list("a", 1, "b", 1, "c", 1, 1, 1, 1, 1)
```

Submission

Submit your mission on the Source Academy.

IMPORTANT: Make sure that everything for your programs to work is on the left hand side (Editor) and not in the Side Content! This is because only the programs in the Editor are preserved when your Avenger grades your submission.