CS2106 AY1920S2 Lab 2 Exercise 1 (demo exercise)

This exercise is the demo exercise, and should be demoed to your lab TA in weeks 5 or 6.

Do One Thing and Do It Well

Part of Unix philosophy is the idea of "Do One Thing and Do It Well", and chaining together programs that do different things to achieve more complicated tasks.

In this exercise, we take that to the extreme.

Suppose we have a simple program that does multiplication well:

```c
// multby.c
#include <stdio.h>

int main(int argv, char *argc[]) {
  if (argv < 2) {
    fprintf(stderr, "Usage: %s <factor>\n", argc[0]);
    return 1;
  }

  size_t a, b;
  sscanf(argc[1], "%zu", &a);
  if (scanf("%zu", &b) != 1) {
    fprintf(stderr, "No input\n");
    return 1;
  }

  printf("%zu", a*b);
  return 0;
}
```

Executing `echo 5 | ./multby 5` returns 25. The pipe `|` causes the shell to redirect the standard output of `echo 5` to the standard input of `./multby 5` using a pipe.

We can use this to compute the factorial if we chain this program like so:

```
echo 1 | ./multby 1 | ./multby 2 | ./multby 3 | ./multby 4 | ./multby 5
```

which outputs `120`, as expected.

Your task

Write a program `chain` to chain N executions of a given program.

- The 1st execution should be given argument 1; the second should be given 2; and so on.
- The standard output of the 1st execution should be piped to the standard input of the 2nd execution; the standard output of the 2nd execution should be piped to the standard input of the 3rd execution; and so on.
- The standard input of the 1st execution, and the standard output of the last execution, should be left as the actual standard input and output respectively.
- `chain` should accept arguments in this format:

  ```
  ./chain <program to chain> <N>
  ```

Executing, for example,

```
./chain ./multby 5
```

should thus be equivalent to

```
./multby 1 | ./multby 2 | ./multby 3 | ./multby 4 | ./multby 5
```

`echo 1 | ./chain ./multby 5` should therefore output `120`, as above.

Syscalls

You may find these syscalls useful:

- `pipe/pipe2`
- `exec` (whichever variant you prefer)
- `close`
- `dup/dup2`
- `fork`
- `wait/waitpid`

Note: calling out to a shell (e.g. via `system`) is not an acceptable solution.