Parsing

This is a **pair** exercise and must be competed in your **tutorial** or **lab** with your partner.

Download `extract.h`, or copy it into your current directory on a CSE system by running

```
$ cp /web/cs1511/17s2/week07/files/extract.h .
```

Download `extract.c`, or copy it into your current directory on a CSE system by running

```
$ cp /web/cs1511/17s2/week07/files/extract.c .
```

In this activity, you will be writing a series of C functions to convert a string to an integer, convert a string to a double, and to extract the `x` , `y` , `z` values from a string of the form "http://almondbread.cse.unsw.edu.au/mandelbrot/2/ `z` / `x` / `y` /tile.bmp".

For example, for the mandelbrot tile centered on (-1.0, -0.2) with a zoom level of 9, the string would be of the form: "http://almondbread.cse.unsw.edu.au/mandelbrot/2/9/-1.0/-0.2/tile.bmp",

The prototypes for these functions are in a file called `extract.h` . Implement these functions in a file called `extract.c` .

You have been given a stub version of `extract.c` , with a simple `main` function already implemented. You have *not* been provided with stub functions; your task is to implement functions for the prototypes specified in `extract.h` :

- `triordinate extract (char *message);`
  `extract` takes a string, and extracts the `x` , `y` , and `z` parts into the `triordinate` structure, defined in *extract.h*.

- `long myAtoL (char *message);`
  `myAtoL` takes a string, extracts a `signed long int` in base 10 from it, and returns that value.

- `double myAtoD (char *message);`
  `myAtoD` takes a string, extracts a `double` in base 10 from it, and returns that value.

When working on `myAtoL` and `myAtoD`, think about the various boundary cases that a number might have. Zeros and signs are some examples… what are some others?

Warning! You may be tempted to use a library function like `sscanf`, `atod`, `atoi`, `atol`, `strtol`, or `strtod`, to read values. For now, don't! Don't use library functions unless you understand them, and for our purposes, that means you should be able to implement it. Try to build functions that actually parse values out of the string.

Your `extract.c` should #include `extract.h` and unit test your functions. We won't test your main, just the three functions from the .h file.

To run Styl-o-matic:

```
$ 1511 stylomatic extract.c
Looks good!
```

You'll get advice if you need to make changes to your code.

Submit your work with the *give* command, like so:

```
$ give cs1511 wk07_parsing
```

Or, if you are working from home, upload the relevant file(s) to the `wk07_parsing` activity on Give Online.