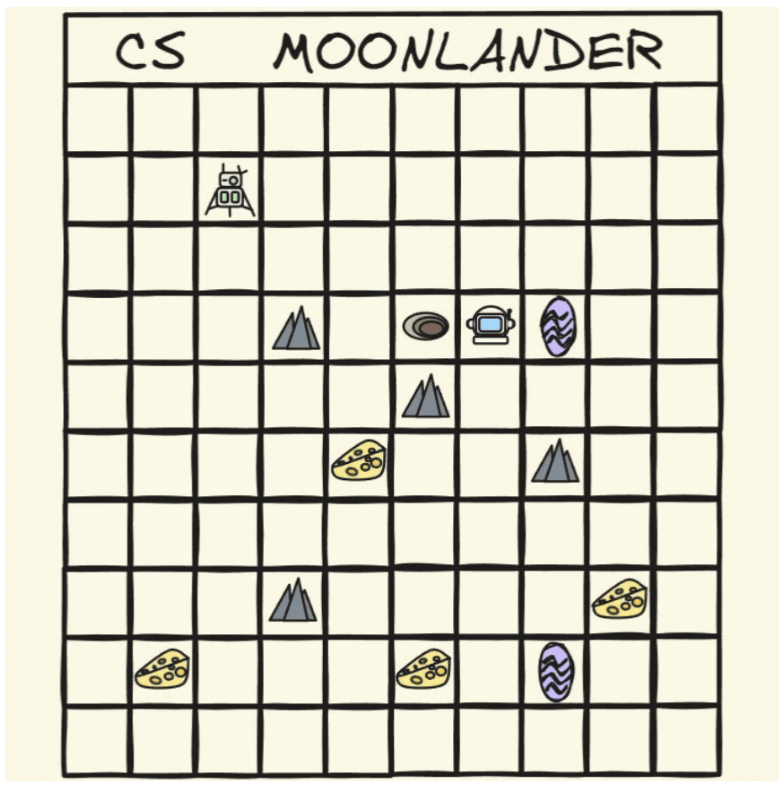
Moonlander

Overview



Moonlander is a small game where the player enters commands to move an astronaut around the moon to collect moon cheese, without running out of oxygen.

The aim of the game is to collect some target amount of cheese and make it back to the lander to export the moon cheese back to Earth! Movement has an oxygen cost and to collect the cheese we may have to navigate around rocks and avoid running out of oxygen completely (a player can drop off cheese and refill their oxygen tank at the lander).

We will begin by setting up the board, then add the ability to move around it to play the game and in the final few stages we will also implement moonquakes and some more interesting methods of travel but we will leave that as a surprise for the moment...

Getting Started

1. Create a new folder for your assignment. Here is an example:

```
$ mkdir ass1
$ cd ass1
```

2. Fetch the starter code using the command below. Alternatively download the starter code here.

```
$ 1091 fetch-activity cs_moonlander
```

3. Check that everything works by running the autotest.

```
$ 1091 autotest cs_moonlander
```

(These should fail initially. If you want to exit running the autotest midway, press `[ctrl-c]`)

Initial Code and Data Structures

The starter code for this assignment includes some functions and defined types:

- A `print_board()` function:
  - This prints out the current state of the board, which ensures that your board printing matches the autotest exactly for a given board.
- An `init_board()` function:
  - This stores a default value in each of the tiles in the provided board.
- A `struct tile` struct:
  - Each square of the board (i.e. the 2D array) holds a `struct tile` , containing the information about what that tile contains.
  - The player's row and col values are stored seperately to the contents of the board
- An `enum entity` enum:
  - This is used in the `struct tile` struct to represent what is on each tile.

Let us look at an example output from the `print_board()` function.

```
+---+---+---+---+---+---+---+---+---+---+
|       C S   M O O N L A N D E R       |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |/|\|   |^^^|   |<(]|   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |0.0|   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |\O/|   |~~~|   |~~~|   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
```

Here, we have tiles containing:

- the lander `/|\`
- a rock `^^^`
- a cheese `<(]`
- a hole down `\O/`
- two portals `~~~`
- the player `0.0`

Where do we store the player?

The player is not stored in the `board` , instead the `print_board()` function takes the player's row and column as arguments whenever we print the board.

- for the above example: `print_board(board, 4, 6);` .

So you can store the player however you like! (e.g. variables for the row/col, or perhaps a struct that contains both the row and the col!)

Reference Implementation

To help you understand how the assignment works, we have written a reference implementation for you to run. You can run it via the command `1091 cs_moonlander` .

```
$ 1091 cs_moonlander
...
```

+ Example Input

FAQ

+ FAQ

Stages

We have broken the assignment spec down into incremental stages:

- **Stage 1:** Place the lander, add cheese & rocks to the board.

- **Stage 2:** Setup the player, add player movement, pick up and drop off cheese, keep oxygen levels correctly updated, refill oxygen at the lander.
- **Stage 3:** Check for a win or lose condition, add a command to quit the program early, add moonquake mechanics.
- **Stage 4:** Make the board 3D, add commands to create holes and jump between levels, add portals, implement travelling through portals.
- **Extension (no marks):** Add a graphical user interface.

Your Tasks

This assignment consists of four stages. Each stage builds on the work of the previous stage, and each stage has a higher complexity than its predecessor. You should complete the stages in order.

Stage 1 ●○○ Stage 2 ●○○ Stage 3 ●●○ Stage 4 ●●● Extension Tools

Stage 1

In stage 1, you will scan and store values to set up the map, including the lander, rock obstacles, and collectible cheese.

There is a reference implementation that you can play by running the command `1091 cs_moonlander` in the terminal.

Here is an example of input for a finished stage 1 that you can try within the reference implementation.

```
$ 1091 cs_moonlander
1 5
c 1 2
c 3 1
r 3 5
r 3 7
R 5 4 8 6
[ctrl+d]
```

Stage 1.1: Placing the Lander

We will begin by scanning in and storing some values from the terminal. We will also need to assert that the values are valid.

In this substage, you will need to do the following:

1. Print the prompt `Please enter the [row] [col] of the lander: `.
2. Scan in the lander's position as `row` and `col`.
3. Check that the row and col values correspond to a valid tile:
   - If the requested tile is not on the board, place the lander at `board[4][4]`.
   - Otherwise, add the lander to the board at the requested tile.
4. Call the provided `print_board()` function with these arguments:
   - `INVALID_INDEX` as the arguments for `player_row` and `player_col`.
   - `BASE_OXY_RATE` as the argument for `oxy_rate`.
   - `0` or `0.0` for all other arguments.

HINT:

You might want to create a function for asserting that a coordinate is valid, this will come up a lot in all 4 stages, including later in stage 1!

Clarifications

- The lander row and col will always be ints.

Examples:

– Example 1.1.1: Adding Lander

Input:

```
6 6
```

Input and Output:

```
$ dcc cs_moonlander.c -o cs_moonlander
$ ./cs_moonlander
Please enter the [row] [col] of the lander: 6 6
+---+---+---+---+---+---+---+---+---+---+
|       C S   M O O N L A N D E R       |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |/|\|   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
```

– Example 1.1.2: Bad Row Input

Input:

```
-3 8
```

Input and Output:

```
$ dcc cs_moonlander.c -o cs_moonlander
$ ./cs_moonlander
Please enter the [row] [col] of the lander: -3 8
+---+---+---+---+---+---+---+---+---+---+
|       C S   M O O N L A N D E R       |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |/|\|   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
```

Autotest

NOTE:

You may like to autotest this section with the following command:

```
1091 autotest-stage 01_01 cs_moonlander
```

Remember to do your own testing!

Stage 1.2: Add cheese and rocks

So far we have only added the lander to the board, that is pretty boring so let us add some moon cheese collectibles and rock obstacles to spice it up!

To do this we will create a setup loop that scans in commands to let the user add `CHEESE` and `ROCK` s to the board. Allow the user to add `CHEESE` using the command in the form `c int int` and add a `ROCK` using the command in the form `r int int` until `[ctrl+d]` is pressed.

In this substage, you will need to do the following:

1. Print the prompt `Please enter cheese and rock locations:\n` .
2. Allow the user to input commands in a loop until `[ctrl+d]` is pressed.
3. Given the command: `c [row] [col]` , add a `CHEESE` to that tile.
4. Given the command: `r [row] [col]` , add a `ROCK` to that tile.
5. After the user enters `[ctrl+d]` , call the provided `print_board()` function with these arguments:
   - INVALID_INDEX as the arguments for `player_row` and `player_col` .
   - BASE_OXY_RATE as the argument for `oxy_rate` .
   - `0` or `0.0` for all other arguments.

Clarifications

- Each command provided will be of the form `char int int` .
- You can not assume that only the `c` or `r` commands will be entered.

Examples

– Example 1.2.1: Adding Cheese to Board

Input:

```
4 3
c 3 7
[ctrl+d]
```

Input and Output:

```
$ dcc cs_moonlander.c -o cs_moonlander
$ ./cs_moonlander
Please enter the [row] [col] of the lander: 4 3
+---+---+---+---+---+---+---+---+---+---+
|       C S   M O O N L A N D E R       |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |/|\|   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
Please enter cheese and rock locations:
c 3 7
[ctrl+d]
+---+---+---+---+---+---+---+---+---+---+
|       C S   M O O N L A N D E R       |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |<(]|   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |/|\|   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
```

– Example 1.2.2: Adding Cheese and Rocks to Board

Input:

```
4 8
c 3 7
r 4 2
c 6 3
r 6 9
[ctrl+d]
```

Input and Output:

```
$ dcc cs_moonlander.c -o cs_moonlander
$ ./cs_moonlander
Please enter the [row] [col] of the lander: 4 8
+---+---+---+---+---+---+---+---+---+---+
|      C S   M O O N L A N D E R      |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |/|\|   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
Please enter cheese and rock locations:
c 3 7
r 4 2
c 6 3
r 6 9
[ctrl+d]
+---+---+---+---+---+---+---+---+---+---+
|      C S   M O O N L A N D E R      |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |<(]|   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |^^^|   |   |   |   |   |/|\|   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |<(]|   |   |   |   |   |^^^|
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
```

Autotest

NOTE:

You may like to autotest this section with the following command:

```
1091 autotest-stage 01_02 cs_moonlander
```

Remember to do your own testing!

Stage 1.3: Add Large Rocks and Check Scan Positioning

Adding rocks one by one can get quite repetitive, so here we will introduce the new `R` command to our handy setup loop from stage 1.2. This will allow us to add large rectangular rocks to the board. We will also print an error message if the cheese and rocks we scan in would be placed on a square that already contains something.

In this substage, you will need to do the following:

1. For our cheese input from stage 1.2, if the requested tile is occupied or not on the board, instead of adding the cheese to the board print `That is not a valid cheese placement!\n` .
2. For our single rock input from stage 1.2, if the requested tile is occupied or not on the board, instead of adding the rock to the board print `That is not a valid rock placement!\n` .
3. Given the command `R [start_row] [start_col] [end_row] [end_col]` , place a large rock with top left corner at `[start_row] [start_col]` and bottom right corner at `[end_row] [end_col]` . A large rock is represented by placing a `ROCK` on every tile in it's area.
   - If any of the following error conditons are true, print `That is not a valid rock placement!\n` and do not add the large rock to the board.
     - Any tile enclosed in the requested rectangle does not lie on the board.
     - Any tile enclosed in the requested rectangle is occupied.
     - `start_row` is greater than `end_row` , or `start_col` is greater than `end_col` .

Clarifications

- A tile is occupied if its `entity` field is not `EMPTY` .

Examples:

– Example 1.3.1: Error Checking Cheese and Single Rocks

Input:

```
3 8
c -1 3
r -1 3
c 3 -1
r 3 -1
c 10 2
r 10 2
c 1 39
r 1 39
[ctrl+d]
```

Input and Output:

```
$ dcc cs_moonlander.c -o cs_moonlander
$ ./cs_moonlander
Please enter the [row] [col] of the lander: 3 8
+---+---+---+---+---+---+---+---+---+---+
|      C S   M O O N L A N D E R      |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |/|\|   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
Please enter the cheese and rock locations:
c -1 3
That is not a valid cheese placement!
r -1 3
That is not a valid rock placement!
c 3 -1
That is not a valid cheese placement!
r 3 -1
That is not a valid rock placement!
c 10 2
That is not a valid cheese placement!
r 10 2
That is not a valid rock placement!
c 1 39
That is not a valid cheese placement!
r 1 39
That is not a valid rock placement!
[ctrl+d]
+---+---+---+---+---+---+---+---+---+---+
|      C S   M O O N L A N D E R      |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |/|\|   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
```

```
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
```

– Example 1.3.2: Adding Large Rocks

Input:

```
2 2
R 4 4 6 6
R 8 2 9 3
[ctrl+d]
```

Input and Output:

```
$ dcc cs_moonlander.c -o cs_moonlander
$ ./cs_moonlander
Please enter the [row] [col] of the lander: 2 2
+---+---+---+---+---+---+---+---+---+---+
|       C S   M O O N L A N D E R       |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |/|\|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
Please enter the cheese and rock locations:
R 4 4 6 6
R 8 2 9 3
[ctrl+d]
+---+---+---+---+---+---+---+---+---+---+
|       C S   M O O N L A N D E R       |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |/|\|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |^^^|^^^|^^^|   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |^^^|^^^|^^^|   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |^^^|^^^|^^^|   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |^^^|^^^|   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |^^^|^^^|   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
```

– Example 1.3.3: Invalid Rock/Cheese Placements

Input:

```
2 2
R 1 2 3 4
R 1 -2 -3 4
R 1 23 3 10
R 8 8 6 6
R 6 6 8 8
[ctrl+d]
```

Input and Output:

```
$ dcc cs_moonlander.c -o cs_moonlander
$ ./cs_moonlander
Please enter the [row] [col] of the lander: 2 2
+---+---+---+---+---+---+---+---+---+---+
|      C S   M O O N L A N D E R      |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |/|\|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
Please enter the cheese and rock locations:
R 1 2 3 4
That is not a valid rock placement!
R 1 -2 -3 4
That is not a valid rock placement!
R 1 23 3 10
That is not a valid rock placement!
R 8 8 6 6
That is not a valid rock placement!
R 6 6 8 8
[ctrl+d]
+---+---+---+---+---+---+---+---+---+---+
|      C S   M O O N L A N D E R      |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |/|\|   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |^^^|^^^|^^^|   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |^^^|^^^|^^^|   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |^^^|^^^|^^^|   |
+---+---+---+---+---+---+---+---+---+---+
|   |   |   |   |   |   |   |   |   |   |
+---+---+---+---+---+---+---+---+---+---+
Player Cheese: 0     Lander Cheese: 0
Oxy: 0.00 / 0.00  @  1.000000 / move
+---+---+---+---+---+---+---+---+---+---+
```

Autotest

NOTE:

You may like to autotest this section with the following command:

```
1091 autotest-stage 01_03 cs_moonlander
```

Remember to do your own testing!

Testing and Submission

Are you finished with this stage? If so, you should make sure to do the following:

- Run `1091 style`, and clean up any issues a human may have reading your code. Don't forget -- **20%** of your mark in the assignment is based on style!
- Autotest for this stage of the assignment by running the `autotest-stage` command as shown below.
- Remember -- *give early, and give often*. Only your last submission counts, but why not be safe and submit right now?

```
1091 style cs_moonlander.c
1091 autotest-stage 01 cs_moonlander
give dp1091 ass1_cs_moonlander cs_moonlander.c
```

Assessment

Assignment Conditions

- **Joint work** is **not permitted** on this assignment.

  This is an individual assignment.

  The work you submit must be entirely your own work. Submission of any work even partly written by any other person is not permitted.

  Except, you may use small amounts (< 10 lines) of general purpose code (not specific to the assignment) obtained from a site such as Stack Overflow or other publicly available resources. You should attribute clearly the source of this code in an accompanying comment.

  Assignment submissions will be examined, both automatically and manually for work written by others.

  Do not request help from anyone other than the teaching staff of DPST1091/CPTG1391, e.g. in the course forum & help sessions.

  Do not post your assignment code to the course forum - the teaching staff can view assignment code you have recently autotested or submitted with give.

  Rationale: this assignment is designed to develop the individual skills needed to produce an entire working program. Using code written by or taken from other people will stop you learning these skills. Other CSE courses focus on the skill needed for work in a team.

- The use of **code-synthesis tools**, such as **GitHub Copilot**, is **not permitted** on this assignment.

  The use of **Generative AI** to generate code solutions is not permitted on this assignment.

  Rationale: this assignment is intended to develop your understanding of basic concepts. Using synthesis tools will stop you learning these fundamental concepts.

- **Sharing, publishing, distributing** your assignment work is **not permitted**.

  Do not provide or show your assignment work to any other person other than the teaching staff of DPST1091/CPTG1391. For example, do not message your work to friends.

  Do not publish your assignment code via the internet. For example, do not place your assignment in a public GitHub repository.

Rationale: by publishing or sharing your work you are facilitating other students using your work which is not permitted. If they submit your work, you may become involved in an academic integrity investigation.

- **Sharing, publishing, distributing your assignment work after the completion of DPST1091/CPTG1391** is **not permitted**.

  For example, do not place your assignment in a public GitHub repository after DPST1091/CPTG1391 is over.

  Rationale: DPST1091/CPTG1391 sometimes reuses assignment themes using similar concepts and content. Students in future terms find your code and use it which is not permitted and you may become involved in an academic integrity investigation.

Violation of the above conditions may result in an academic integrity investigation with possible penalties, up to and including a mark of 0 in DPST1091/CPTG1391 and exclusion from UNSW.

Relevant scholarship authorities will be informed if students holding scholarships are involved in an incident of plagiarism or other misconduct. If you knowingly provide or show your assignment work to another person for any reason, and work derived from it is submitted you may be penalised, even if the work was submitted without your knowledge or consent. This may apply even if your work is submitted by a third party unknown to you.

Note, you will not be penalised if your work is taken without your consent or knowledge.

For more information, read the UNSW Student Code, or contact the course account. The following penalties apply to your total mark for plagiarism:

0 for the assignment	Knowingly providing your work to anyone and it is subsequently submitted (by anyone).
0 for the assignment	Submitting any other person's work. This includes joint work.
0 FL for DPST1091	Paying another person to complete work. Submitting another person's work without their consent.

Submission of Work

You should submit intermediate versions of your assignment. Every time you autotest or submit, a copy will be saved as a backup. You can find those backups here, by logging in, and choosing the yellow button next to 'cs_moonlander.c'.

Every time you work on the assignment and make some progress you should copy your work to your CSE account and submit it using the `give` command below.

It is fine if intermediate versions do not compile or otherwise fail submission tests.

Only the final submitted version of your assignment will be marked.

You submit your work like this:

```
$ give dp1091 ass1_cs_moonlander cs_moonlander.c
```

Assessment Scheme

This assignment will contribute 20% to your final mark.

80% of the marks for this assignment will be based on the performance of the code you write in `cs_moonlander.c`

20% of the marks for this assignment will come from manual marking of the readability of the C you have written. The manual marking will involve checking your code for clarity, and readability, which includes the use of functions and efficient use of loops and if statements.

Marks for your performance will be allocated roughly according to the below scheme.

100% for Performance	Completely Working Implementation, which exactly follows the spec (Stage 1, 2, 3 and 4).
85% for Performance	Completely working implementation of Stage 1, 2 and 3.
65% for Performance	Completely working implementation of Stage 1 and Stage 2.
35% for Performance	Completely working implementation of Stage 1.

The Challenge stage of the assignment is NOT worth any marks, but is something fun for you to work on getting to know a new library and building something more visual!

Style Marking Rubric

	0	1	2	3	4
Formatting (/5)					
Indentation (/2) - Should use a consistent indentation scheme.	Multiple instances throughout code of inconsistent/bad indentation	Code is mostly correctly indented	Code is consistently indented throughout the program		
Whitespace (/1) - Should use consistent whitespace (for example, 3 + 3 not 3+ 3)	Many whitespace errors	No whitespace errors			
Vertical Whitespace (/1) - Should use consistent whitespace (for example, vertical whitespace between sections of code)	Code has no consideration for use of vertical whitespace	Code consistently uses reasonable vertical whitespace			
Line Length (/1) - Lines should be max. 80 characters long	Many lines over 80 characters	No lines over 80 characters			
Documentation (/5)					
Comments (incl. header comment) (/3) - Comments have been used throughout the code above code sections and functions to explain their purpose. A header comment (with name, zID and a program description) has been included	No comments provided throughout code	Few comments provided throughout code	Comments are provided as needed, but some details or explanations may be missing causing the code to be difficult to follow	Comments have been used throughout the code above code sections and functions to explain their purpose. A header comment (with name, zID and a program description) has been included	
Function/variable/constant naming (/2) - Functions/variables/constants names all follow naming conventions in style guide and help in understanding the code	Functions/variables/constants names do not follow naming conventions in style guide and help in understanding the code	Functions/variables/constants names somewhat follow naming conventions in style guide and help in understanding the code	Functions/variables/constants names all follow naming conventions in style guide and help in understanding the code		
Organisation (/5)					
Function Usage (/4) - Code has been decomposed into appropriate functions separating functionalities	No functions are present, code is one main function	Some code has been moved to functions	Some code has been moved to sensible/thought out functions, and/or many functions exceed 50 lines (incl. main function)	Most code has been moved to sensible/thought out functions, and/or some functions exceed 50 lines (incl. main function)	All code h been meaningfu decompos into functions a maximu of 50 lines (incl. The main function)
Function Prototypes (/1) - Function Prototypes have been used to declare functions above main	Functions are used but have not been prototyped	All functions have a prototype above the main function or no functions are used			
Elegance (/5)					

Overdeep nesting (/2) - You should not have too many levels of nesting in your code (nesting which is 5 or more levels deep)	Many instances of overdeep nesting	<= 3 instances of overdeep nesting	No instances of overdeep nesting	
Code Repetition (/2) - Potential repetition of code has been dealt with via the use of functions or loops	Many instances of repeated code sections	<= 3 instances of repeated code sections	Potential repetition of code has been dealt with via the use of functions or loops	
Constant Usage (/1) - Any magic numbers are #defined	None of the constants used throughout program are #defined	All constants used are #defined and are used consistently in the code		
Illegal elements				
Illegal elements - Presence of any illegal elements indicated in the style guide	**CAP MARK AT 16/20**			

Allowed C Features

In this assignment, there are no restrictions on C Features, except for those in the style guide. If you choose to disregard this advice, you **must** still follow the style guide.

You also may be unable to get help from course staff if you use features not taught in DPST1091. Features that the Style Guide identifies as illegal will result in a penalty during marking. You can find the style marking rubric above.

Due Date

This assignment is due **Week 8 Friday 09:00am** (2025-03-14 09:00:00). For each day after that time, the maximum mark it can achieve will be reduced **by 5%** (off the ceiling). For example at:

- Less than 1 day (24 hours) past the due date, the maximum mark you can get is **95%**.
- Less than 2 days (48 hours) past the due date, the maximum mark you can get is **90%**.
- Less than 5 days (120 hours) past the due date, the maximum mark you can get is **75%**.

No submissions will be accepted at 5 days late, unless you have special provisions in place.

Change Log

Version 1.0 (2025-02-21 09:00)	• Assignment Released

DPST1091/CPTG1391 25T1: Programming Fundamentals!