Pokémon: Gotta Find Them All!

Assignment 1
CSSE1001/7030
Semester 1, 2020
10 marks

Due Date: 27th March, 17:00

1 Introduction

The goal of this assignment is to produce a text-based game with Python.

2 Getting Started

To start, download a1.zip from Blackboard and extract the contents. The a1.zip folder contains all the necessary files to start this assignment. Some support code has been provided to assist with implementing the tasks. You will be required to implement your assignment in *a1.py*.

The other provided file is *a1_support.py*, which contains some code to help you implement your assignment. You are not compelled to use this file to implement your assignment but it is strongly recommended. Do not make changes to the *a1_support.py* file. The only file that you are required to submit is *a1.py*. It could cause unexpected errors if you made changes to the *a1_support.py* file as well.

3 Concepts

Pokémon: Gotta Find Them All! is a single-player puzzle game where the objective is for the player to find all the Pokemon without scaring them away by stepping on them. When the game starts the player should first be prompted for the game "size" (i.e. the grid size), before being prompted for the number of Pokemon used in the game. The game should then be displayed with all tiles unexposed (see Fig. 1 for a sample game of grid size, 7).

At each turn the player should be prompted for an action. Table 1 shows a list of valid actions:

Input	Action	Example
'Upper Case Character–number-'	Select a cell	'A1'
'f -Upper Case Character–number-'	Place/remove a flag	'f A1'
'h'	Help text (see examples at end)	'h'
'q'	Quit	'q'
':)'	Restart	':)'

Table 1: List of valid actions.

The list of characters used in the game is shown in Table 2.

Character	**Description**
'|'	WALL_VERTICAL
'-'	WALL_HORIZONTAL
'☺'	POKEMON
'♥'	FLAG
'∼'	UNEXPOSED cell
'0'	EXPOSED cell (this could also be any number from 0-8)

Table 2: List of characters used in the game.

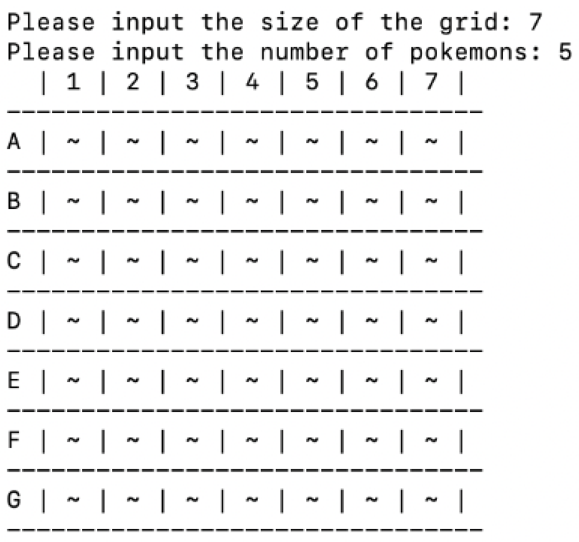
If, say, 'C4' is entered, then the cell at position, C4, should be uncovered to reveal a number that indicates how many Pokemon are hidden in the neighbouring cells. If no Pokemon are present in any neighbouring cells then the number 0 should be displayed. For cells which have a zero value (i.e. no neighbouring pokemons) all the cell's neighbours are uncovered (i.e. they are made to display the number of pokemons in neighbouring cells). If one of those neighbouring cells is also zero then all of that cell's neighbours are also uncovered. This process repeats until there are no more 0's uncovered. This forms a **section**. (See Fig. 2 for an illustration of this). (The big fun search function in the support code will search through the game to determine which cells needs to be uncovered)

If, say, 'f B1' is entered, then a flag will be placed on the cell at position, B1, provided there is no flag currently on that cell. Alternatively, if there is a flag on that cell, the flag will be removed.

If 'h' is entered, then advice about how to play the game is provided. See the Section, "Examples of how the game should run", to see what this text should be. If 'q' is entered, then the game provides a prompt to make sure the player wants the game to end. If they do, then the game ends. (See the section, "Examples of how the game should run" to see the required text). If ':)' is entered then the game is reset (i.e. restarted).

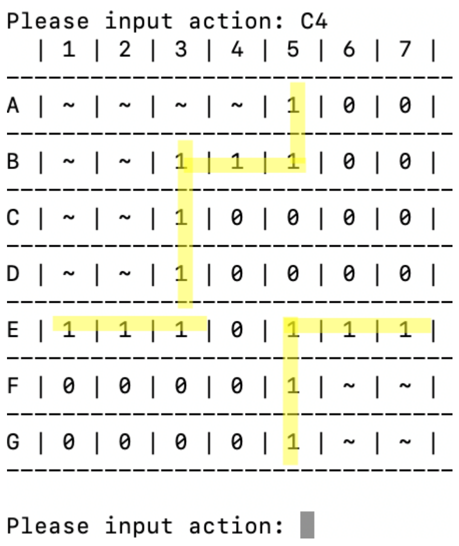
The game ends when either:

- The player loses the game by selecting a cell that contains a Pokemon, thus scaring away all the Pokemon (see Fig. 3).
- The player wins the game by correctly flagging all cells with hidden Pokemon **AND** exposing all cells that do not contain Pokemon.



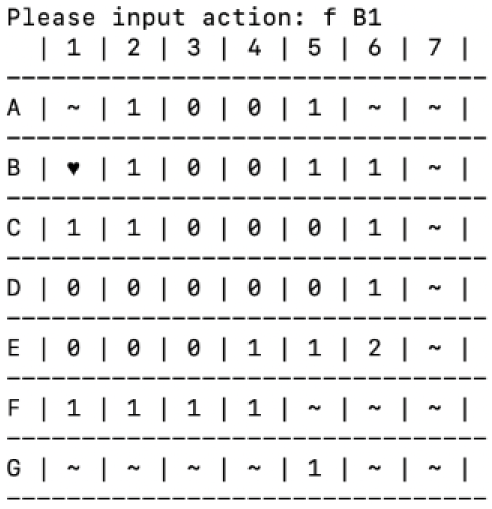
```
Please input the size of the grid: 7
Please input the number of pokemons: 5
  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
--------------------------------
A | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------
B | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------
C | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------
D | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------
E | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------
F | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------
G | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------
```

Figure 1: Example of the start of the game.



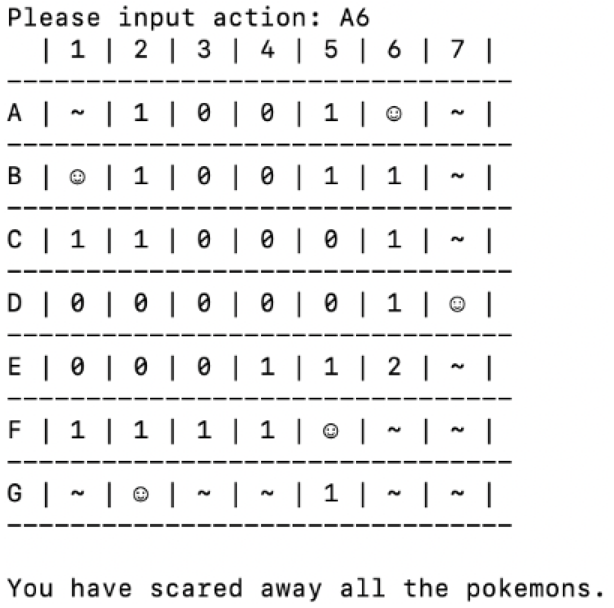
```
Please input action: C4
  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
--------------------------------
A | ~ | ~ | ~ | ~ | 1 | 0 | 0 |
--------------------------------
B | ~ | ~ | 1 | 1 | 1 | 0 | 0 |
--------------------------------
C | ~ | ~ | 1 | 0 | 0 | 0 | 0 |
--------------------------------
D | ~ | ~ | 1 | 0 | 0 | 0 | 0 |
--------------------------------
E | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
--------------------------------
F | 0 | 0 | 0 | 0 | 1 | ~ | ~ |
--------------------------------
G | 0 | 0 | 0 | 0 | 1 | ~ | ~ |
--------------------------------

Please input action: 
```

Figure 2: Example of selecting a cell with 0 surrounding Pokemons. (Neighbouring cells have to be repeatedly uncovered until there are no more neighbouring cells with 0 Pokemons. Effectively, a border of non-zero values forms around the original cell. This forms a **section**.



```
Please input action: f B1
  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
-------------------------------
A | ~ | 1 | 0 | 0 | 1 | ~ | ~ |
-------------------------------
B | ♥ | 1 | 0 | 0 | 1 | 1 | ~ |
-------------------------------
C | 1 | 1 | 0 | 0 | 0 | 1 | ~ |
-------------------------------
D | 0 | 0 | 0 | 0 | 0 | 1 | ~ |
-------------------------------
E | 0 | 0 | 0 | 1 | 1 | 2 | ~ |
-------------------------------
F | 1 | 1 | 1 | 1 | ~ | ~ | ~ |
-------------------------------
G | ~ | ~ | ~ | ~ | 1 | ~ | ~ |
-------------------------------
```

Figure 3: Example of flagging the cell B1.



```
Please input action: A6
  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
-------------------------------
A | ~ | 1 | 0 | 0 | 1 | ☺ | ~ |
-------------------------------
B | ☺ | 1 | 0 | 0 | 1 | 1 | ~ |
-------------------------------
C | 1 | 1 | 0 | 0 | 0 | 1 | ~ |
-------------------------------
D | 0 | 0 | 0 | 0 | 0 | 1 | ☺ |
-------------------------------
E | 0 | 0 | 0 | 1 | 1 | 2 | ~ |
-------------------------------
F | 1 | 1 | 1 | 1 | ☺ | ~ | ~ |
-------------------------------
G | ~ | ☺ | ~ | ~ | 1 | ~ | ~ |
-------------------------------

You have scared away all the pokemons.
```

Figure 4: Selecting cell A6 which has a hidden Pokemon.

3.1 SOME PROGRAM ENTITIES:

`game`: This is a string which represents the cells in the game. This should initially be a string entirely made of UNEXPOSED characters (i.e. entirely of ∼'s).

`index`: This is an integer which indicates the position in a string (usually the game string).

`grid_size`: This is an integer which represents the size of the grid. i.e. a `grid_size` of 7 corresponds to a game that has 7 rows x 7 columns.

`position`: Tuple representing a row, column position of a cell.

`pokemon_locations`: This is a tuple containing all the Pokemon locations (i.e. indexes within the game string). The number of elements within this tuple should be specified at the start of the game by prompting the player to enter how many Pokemon they wish to have in the game. The support code contains a function which generates the Pokemon locations.

`Directions`: Directions to neighbouring cells are: ”up”, “down”, “left”, “right”, “up-left”, “up-right”, “down-left”, “down-right”. See `DIRECTIONS` in the support code.

4 Implementation

The following functions must be written when implementing the game. (Additional functions are allowed if necessary)

`display_game(game, grid_size)`
This function prints out a grid-shaped representation of the game, given the game string and the grid size as arguments.

`parse_position(action, grid_size) -> tuple<int, int>`
This function checks if the input action is in a valid format. i.e it checks if the entered action fits the criteria in Table 1. If a non-valid action is entered, None should be returned. If the action is “Select a cell” or “Flag/remove a cell”, the position of the cell should be returned as a tuple. E.g. A1 should return the tuple, (0, 0).

	1	2	3
A	(0,0)	(1,0)	(2,0)
B	(0,1)	(1,1)	(2,1)
C	(0,2)	(1,2)	(2,2)

Table 3: Positions in grid and corresponding tuples.

`position_to_index(position, grid_size) -> int`
This function should convert the row, column coordinate in the grid to the game strings index. The function returns an integer representing the index of the cell in the game string.

`replace_character_at_index(game, index, character) -> str`
This function returns an updated game string with the specified character placed at the specified index.

`flag_cell(game, index) -> str`
This function returns an updated game string after “toggling” the flag at the specified index in the game string.

`index_in_direction(index, grid_size, direction) -> int`
This function takes in the index to a cell in the game string and returns a new index corresponding to an adjacent cell in the specified direction.

For example:

	1	2	3
A	∼	∼	☺
B	∼	∼	∼
C	∼	♥	∼

= Corresponds to =

index	0	1	2	3	4	5	6	7	8
char	∼	∼	☺	∼	∼	∼	∼	♥	∼

If the input index is **5**, and the direction specified is **"up"** then the updated index is **2**.

`neighbour_directions(index, grid_size) -> list<int, ...>`
This function returns a list of indexes that have a neighbouring cell. (Note that the cells at the edges of the grid do not have all possible directions).

`number_at_cell(game, pokemon_locations, grid_size, index) -> int`
This function returns the number of Pokemon in neighbouring cells.

`check_win(game, pokemon_locations) -> bool`
This function returns True if the player has won the game, and returns False otherwise.

`main()`
This function handles player interaction. At the start of the game the player should be prompted with the following:

1. "Please input the size of the grid: " where the player has to enter a single integer (26 being the maximum) representing the size. Followed by
2. "Please input the number of pokemons: " where the player has to enter a single integer representing how many pokemons they would like to be in the game.

The game should generate the correct number of pokemon (hint: generate_pokemons) then loop continuously until either the game finishes or the player quits.

Note: The `big_fun_search` function is required to uncover a **section** of cells.

5 Examples of the game running

```
Please input the size of the grid:  5
Please input the number of pokemons:  5
| 1 | 2 | 3 | 4 | 5 |
------------------------
A | ~ | ~ | ~ | ~ | ~ |
------------------------
B | ~ | ~ | ~ | ~ | ~ |
------------------------
C | ~ | ~ | ~ | ~ | ~ |
------------------------
D | ~ | ~ | ~ | ~ | ~ |
------------------------
E | ~ | ~ | ~ | ~ | ~ |
------------------------

Please input action:  E5
| 1 | 2 | 3 | 4 | 5 |
------------------------
A | ~ | ~ | ~ | ~ | ~ |
------------------------
B | ~ | ~ | ~ | ~ | ~ |
------------------------
C | ~ | ~ | 1 | 1 | 1 |
------------------------
D | ~ | ~ | 1 | 0 | 0 |
------------------------
E | ~ | ~ | 1 | 0 | 0 |
------------------------

Please input action:  A1
| 1 | 2 | 3 | 4 | 5 |
------------------------
A | 1 | ~ | ~ | ~ | ~ |
------------------------
B | ~ | ~ | ~ | ~ | ~ |
------------------------
C | ~ | ~ | 1 | 1 | 1 |
------------------------
D | ~ | ~ | 1 | 0 | 0 |
------------------------
E | ~ | ~ | 1 | 0 | 0 |
------------------------

Please input action:  A5
| 1 | 2 | 3 | 4 | 5 |
------------------------
A | 1 | ~ | ~ | ~ | 1 |
------------------------
B | ~ | ~ | ~ | ~ | ~ |
------------------------
C | ~ | ~ | 1 | 1 | 1 |
------------------------
```

```
D | ~ | ~ | 1 | 0 | 0 |
------------------------
E | ~ | ~ | 1 | 0 | 0 |
------------------------

Please input action:  E1
| 1 | 2 | 3 | 4 | 5 |
------------------------
A | 1 | ~ | ~ | ~ | 1 |
------------------------
B | ~ | ~ | ~ | ~ | ~ |
------------------------
C | ~ | ~ | 1 | 1 | 1 |
------------------------
D | ~ | ~ | 1 | 0 | 0 |
------------------------
E | 1 | ~ | 1 | 0 | 0 |
------------------------

Please input action:  C1
| 1 | 2 | 3 | 4 | 5 |
------------------------
A | 1 | ~ | ~ | ~ | 1 |
------------------------
B | ~ | ☺ | ~ | ~ | ☺ |
------------------------
C | ☺ | ~ | 1 | 1 | 1 |
------------------------
D | ~ | ~ | 1 | 0 | 0 |
------------------------
E | 1 | ☺ | 1 | 0 | 0 |
------------------------

You have scared away all the pokemons.



Please input the size of the grid:  10
Please input the number of pokemons:  5
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
--------------------------------------------
A | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
B | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
C | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
D | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
E | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
F | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
```

```
G | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
H | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
I | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
J | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------

Please input action:  h
h - Help.
<Uppercase Letter><number> - Selecting a cell (e.g.  'A1')
f <Uppercase Letter><number> - Placing flag at cell (e.g.  'f A1')
:)  - Restart game.
q - Quit.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
--------------------------------------------
A | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
B | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
C | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
D | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
E | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
F | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
G | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
H | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
I | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
J | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------

Please input action:  q
You sure about that buddy?  (y/n):  n
Let's keep going.
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
--------------------------------------------
A | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
B | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
C | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
D | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
E | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
--------------------------------------------
```

```
F | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
-------------------------------------------
G | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
-------------------------------------------
H | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
-------------------------------------------
I | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
-------------------------------------------
J | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ | ~ |
-------------------------------------------

Please input action:  A1
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
-------------------------------------------
A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ~ |
-------------------------------------------
D | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
E | 0 | 1 | ~ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
F | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
-------------------------------------------
G | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ~ | 2 | 1 |
-------------------------------------------
H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | ~ |
-------------------------------------------
I | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
J | 0 | 1 | ~ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------

Please input action:  a1
That ain't a valid action buddy.
Please input action:  f C10
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
-------------------------------------------
A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ♥ |
-------------------------------------------
D | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
E | 0 | 1 | ~ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
F | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
-------------------------------------------
G | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ~ | 2 | 1 |
-------------------------------------------
```

```
H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | ~ |
-------------------------------------------
I | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
J | 0 | 1 | ~ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------

Please input action:  f E3
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
-------------------------------------------
A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ♥ |
-------------------------------------------
D | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
E | 0 | 1 | ♥ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
F | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
-------------------------------------------
G | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ~ | 2 | 1 |
-------------------------------------------
H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | ~ |
-------------------------------------------
I | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
J | 0 | 1 | ~ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------

Please input action:  f G8
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
-------------------------------------------
A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ♥ |
-------------------------------------------
D | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
E | 0 | 1 | ♥ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
F | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
-------------------------------------------
G | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ♥ | 2 | 1 |
-------------------------------------------
H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | ~ |
-------------------------------------------
I | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
J | 0 | 1 | ~ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
```

```
Please input action:  f H10
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
-------------------------------------------
A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ♥ |
-------------------------------------------
D | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
E | 0 | 1 | ♥ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
F | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
-------------------------------------------
G | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ♥ | 2 | 1 |
-------------------------------------------
H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | ♥ |
-------------------------------------------
I | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
J | 0 | 1 | ~ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------

Please input action:  f J3
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
-------------------------------------------
A | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
B | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
C | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ♥ |
-------------------------------------------
D | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
E | 0 | 1 | ♥ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------
F | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 |
-------------------------------------------
G | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ♥ | 2 | 1 |
-------------------------------------------
H | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 2 | ♥ |
-------------------------------------------
I | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
-------------------------------------------
J | 0 | 1 | ♥ | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
-------------------------------------------

You win.
>>>
```

If the player decides to quit the game (q command followed by y command) the following should be displayed:
`Catch you on the flip side.`

If the player wishes to restart the game (:) command) the following should be displayed:

```
It's rewind time.
```

Marking

6 Functionality Assessment

The functionality will be marked out of 7. Your assignment will be put through a series of tests and your functionality mark will be proportional to the number of tests you pass. If, say, there are 25 functionality tests and you pass 20 of them, then your functionality mark will be $20/25 * 7$.

You will be given about 90% of the functionality tests before the due date for the assignment so that you can gain a good idea of the correctness of your assignment yourself before submitting. You should, however, make sure that your program meets all the specifications given in the assignment. That will ensure that your code passes all the tests, including the ones you are not given before the due date.

Note: Functionality tests are automated, therefore, string outputs need to exactly match what is expected.

7 Code Style

The style of your assignment will be assessed by one of the tutors, and you will be marked according to the style rubric provided with the assignment. The style mark will be out of 3.

8 Feedback Session

In the week following the assignment, students will be able to go to a prac session and request feedback on their assignment from a tutor. They will not be able to get assignment marks, simply feedback. These feedback sessions are not compulsory.

You should go to the prac session that you are enrolled in to get your feedback. If the prac sessions are not too busy then you can request that a tutor give you feedback even if you are not enrolled in that session.

9 Assignment Submission

Your assignment must be submitted via the Assignment 1 submission link on Blackboard. You must submit a Python file containing your implementation of the assignment. Late submission of the assignment will not be accepted. Do not wait until the last minute to submit your assignment, as the time to upload it may make it late. Multiple submissions are allowed, so ensure that you have submitted an almost complete version of the assignment well before the submission deadline. Your latest on-time, submission will be marked. Ensure that you submit the correct version of your assignment. In the event of exceptional circumstances, you may submit a request for an extension. See the course profile for details of how to apply for an extension. Requests for extensions must be made no later than 48 hours prior to the submission deadline. The expectation is that with less than 48 hours before an assignment is due it should be substantially completed and a1.py submittable. Applications for extension, and any supporting documentation (e.g. medical certificate), must be submitted via my.UQ. You must retain the original documentation for a minimum period of six months to provide as verification should you be requested to do so.