Cardiff University
School of Computer Science & Informatics

CM1101

LAB EXERCISE: BEGINNINGS OF A GAME

2016

INSTRUCTIONS: This set of exercises will guide you towards building your own text-based adventure game, step by step. Initially, we will concentrate only on the bare essentials: movement of the protagonist around the world, and we will continue to build up the game in the subsequent labs. If you do not manage to finish all the exercises in the lab, please continue doing them at the next lab or at home. Remember, the lab tutors are here to help. If you get stuck — do not be shy, raise your hand and ask for advice.

Good luck!

1. Download `game1_template.zip` from Learning Central and unzip it into a folder on your `H:` drive. This template provides a basic structure for the game. The map of the game and the descriptions of all rooms are found in `map.py` file, and the main code for the game is in `game.py`. These are just templates which you need to fill in with your code. Each function is commented and some `doctest` tests are provided. Feel free to examine the structure of the game code.

2. Edit `map.py`, in which the rooms are defined as dictionaries, and complete the `"exits"` field for each room according to this map:

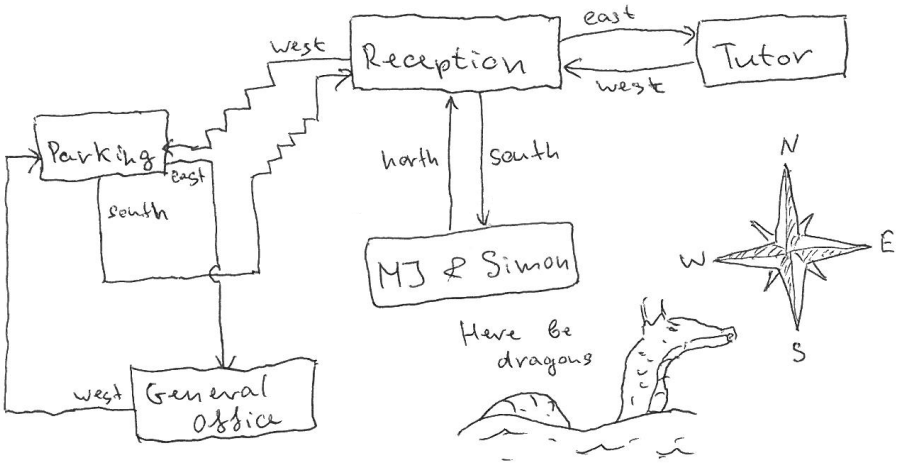

Figure 1: Map of the game.

The `"exits"` field is itself a dictionary, where the keys are the names of the exits (*e.g.* `"south"`) and values are identifiers of the rooms to which these exits lead (*e.g.* `"MJ and Simon"`). See the `rooms` dictionary at the bottom of `map.py` for the list of room identifiers (this dictionary maps room identifiers to rooms themselves). One of the exits (leading `"south"` from `"Reception"` to `"MJ and Simon"`) has been defined for you. Complete the rest.

3. Examine the `main()` function which contains the main game loop. The way it works is as follows:

   - It prints the name and description of the current room (which at start is initialised to `"Reception"`).
   - It displays a menu of available exits and prompts the player where they want to go next.
   - The current room is updated to reflect the movement of the player.

   This loop repeats forever. Make sure you understand the main game logic here before proceeding.

4. Test your template by running:

   ```
   python -m doctest -v game.py
   ```

   You should see a test report which contains mostly failed tests. As you fill in the provided template with your code, more and more tests will pass and eventually, towards the end of the lab, all tests should pass. If you want to see the output *only* for the tests that failed, use:

   ```
   python -m doctest game.py
   ```

   Proceed by completing all the functions in the provided template. Make sure to run tests each time to verify the correctness of your code.

5. Complete the `display_room` function, for which the description and some tests have been provided in the comments. Make sure to read these carefully. Run the tests again to make sure that your implementation is correct.

6. Complete the function `print_menu_line` which simply prints a line in a menu of exits, given the name of the exit and where it leads to. Again, read the documentation for this function and test examples carefully, and remember to run the tests.

7. Complete the function `print_menu` which, given a list of available exits, prints all of them (use the previously completed `print_menu_line` function to print information about each exit). You also need to define a helper function `exit_leads_to` which returns the name of the room to which a particular exit leads. Again, run tests.

8. Complete the function `menu` which displays a list of available exits and prompts the player where they want to go next. For now, assume that the player always enters a valid exit (*i.e.* no normalisation of input and checking for correctness is required).

9. Complete the function `move` which updates the current room based on the exit the player chose in the `menu` function. The game should now be basically playable (if we assume the player always enters correct commands). Run the tests and try playing your game.

10. To check whether the player has selected a valid exit from the menu, complete the function `is_valid_exit`. Now, update your `menu` function to repeat asking the player until they enter a valid exit.

11. To make processing of the input more robust, and to allow the player some flexibility, *e.g.* `South!` or `sOuTh...`, we define the function `normalise_input` which removes punctuation and whitespace, and converts the input to lower case. Complete this function, having completed the helper functions `remove_punct` and `remove_spaces`. You may dig into the Python standard library and get away with easy solutions (for example, `.strip()` will remove the whitespace around a string), however it is more instructive to write these functions yourself!
