**Assignment 2**: adventure

**Author**: Zifan Lin (zl433)

**Summary**

The most challenging part of this assignment is to choose and design the most appropriate data structure to represent the game state and its parts. I used set, map, list, and some custom data structures and lots of conversions need to be done among them, so writing helper functions to do all the conversions was challenging as well.

Some major design decisions I made include: separate commands that change and do not change game state into two different functions do’ and do\_static; create two new files convert.ml and validate.ml for the parsing functions, converting functions, and functions that validate whether a command is valid; make error message as part of the command type, so the same type is returned to Main no matter the command is valid or not.

When developing, I have encountered bugs like can’t pick up items with capital letters in their ids, throwing Not\_found exceptions when going to exits that exist for sure, but they have all been fixed, so there is no known problem. One thing I could do to improve my code is to add more sanity checks to the existing three to spot more discrepancies in game file before player starts playing.

Because my own adventure is quite long, to save time for grader, please refer to my game design: <https://drive.google.com/file/d/1zGcUM96JhZvMwxfav3WImiwHdKIaCNX3/view?usp=sharing>

**Design, Implementation, and Testing**

I designed custom types for room, item, description, and exit. The fields of these types are basically the same as the fields in JSON. I also designed custom types for state and command and detailed description of their fields can be found in command.ml and state.ml. To maintain information about rooms, items, and item locations, I created three modules RoomSet, ItemSet (both are created using the functor Set.Make), and LocationMap.

My helper functions are divided into the following groups: readers (in Convert), JSON parsers (in Convert), sanity checkers (in Convert), helpers that convert among data types (in Convert), command validators (in Validate), and some helpers in State that do not fall in these categories. Examples of important helper functions include is\_sublist (which is very helpful in determining which description to print and which exit is locked), and most of the conversion functions.

This assignment was mostly done bottom-up. I wrote on a piece of paper about what files and functions I would need, wrote each file separately, and finally combined them all together to test and debug (each separate file was tested in utop beforehand). For testing, I used both OUnit test and manual testing. My OUnit test cases cover all commands and all fields of game state, whether the command is supposed to change it or not, so I believe the test cases are rather comprehensive. For manual testing, I would use string\_of\_state to print out the game state, type in all sorts of commands to see how the state changes and when exceptions occur, make corrections, and try again. I used oneroom and threerooms to find simple bugs, and gates and my own game file to find bugs that are more deeply hidden.

**Division of labor**

I wrote all code myself (of course).

I would like to thank Bolin Huang for helping me to test the game engine and my own adventure, and for providing helpful comments and criticisms.

**Known problems**

None.

**Comments**

I spent approximately 25 hours on designing and building the game engine and a few more on testing. “Plan of attack” in the assignment writeup was hugely helpful but it could be even more helpful if there is a flowchart showing which file produces what and passes it to which other file.

The hardest and most surprising part of this assignment is I had to build lots of custom data structures and do lots of conversions between them. And once a decision choice is made, changing it means changing a significant fraction of the code written, so this assignment really emphasizes careful design before implementation.

The thing I like (and hate) the most about this assignment is the huge freedom I was given. Most of the CS assignments I have done give me a basic structure and ask me to implement a few functions, but in A2 I need to figure out the structure myself, so it really makes me have some hard times. But after going through the hardship, my way of thinking about programming assignments is changed, and I begin to enjoy this kind of assignment.