**Spike:** Task 12

**Title:** Game Graphs from Data

**Author:** Sam Huffer, 101633177

# Goals / deliverables:

* The specification for a text-file format for representing (at this stage) the details of a world, its locations, and the connections between them.
* Code that can:
  + Load the world from a text file from the Select Adventure stage
  + Store the world data in your program.
  + Allow players to move between locations in the world, with more directions than just “North”, “East”, etc.

# Technologies, Tools, and Resources used:

* Visual Studio 2019
* Microsoft Word

# Tasks undertaken

* I copied the spike report template into the task folder, stripping out the original content and replacing it with goals and resources pertaining to the task at hand.
* I created a blank Visual Studio project “Zorkish Adventure” and copied in all the source files used in Task 10: Game Data Structures, so that for future tasks I wouldn’t have any folders or files named for previous tasks to worry about renaming.
* I wrote out the specification for recording the details of a world, its locations and its items in a text file that could be loaded into the game, as well as how to format a list of available worlds and their text files.
* I designed a basic world with a room for each cardinal direction, one in the middle of all of them, a cellar under the north room, an attic above the east room, and a void that one enters from the west room and exits from into the south room. I then typed up the details of the world in a text file, adding the items used in the previous void test world to the specification. Next, I added a text file for listing all available worlds, listing the test world in it.
* I reconfigured SelectAdventure.Setup() to read in the text file “Worlds.txt” to ascertain the worlds available to the player and print them out instead of a hard-coded list of worlds.

# Text-File Format

### List of Worlds (Worlds.txt)

The list of worlds formatting is very simple. Each line is a world, listing its name and filename, separating the two with a “:”. For example:

“World Name:world\_filename.txt”

would have “World Name” displayed in the SelectAdventure stage, while “world\_filename.txt” would be passed to the constructor of the World class to draw its specification from there.

### Worlds (e.g. “Test World.txt”)

The formatting for a world’s details are a bit trickier. Again, different details are separated by “:”’s, but here different pieces of information are given prefixes to identify what they entail:

* #Comment ==> Ignore
* *(blank line; ignore)*
* W:World Name
* L:location\_id:location\_name:location description
* C:container\_item\_id:container\_item\_name:container item description:container\_id

*(Note: container\_id is what it resides within, whether a location or another container item.)*

* I:item\_id:item\_name:item description:container\_id
* P:location\_from\_id:direction:location\_id\_to:path description

*(Note: this lists the connections or pathways that a location (location\_from\_id) has to other locations (location\_to\_id), which are stored in the <string, string> map Location.neighbours, hence the prefix “N”.)*

* S:starting\_location\_id

# What we found out