

Adventure Game Assignment

This is a scaffolded assignment using some skeleton code to complete.
You will be using these concepts covered so far:

1. functions
2. if then elseif else conditional statements
3. while or repeat until loop
4. for loop
5. tables

This is a possible output:

```
C:\Program Files (x86)\Lua\lua.exe
SCHOOL OF DOOM

Welcome to the most epic of adventures
You are trapped inside the school from Hell!

See if you can escape.....

Type the name of your player_fred
Choose your character
  1) Genius
  2) Average Joe
  3) Teacher
  4) Earthworm
Type the number of your choice (1 to 4)_3
Choose your weapon
  1) feather
  2) axe
  3) plastic fork
  4) nuclear warhead
Type the number of your choice (1 to 4)_4
Player properties
Name: Fred
Character: Teacher
Weapon: nuclear warhead
Health: 45
Strength: 55
You are in dungeon A rat infested cell
In this area there is key, torch,
Which way do you want to go?
  1) office
  2) kitchen
  3) Quit
Type the number of your choice (1 to 3)_2
You are in kitchen A steamy industrial kitchen in MacDonalds
In this area there is knife,
Which way do you want to go?
  1) dungeon
  2) Quit
Type the number of your choice (1 to 2)_2
Thank you for playing
Press any key to exit
```

Copy the code from Github or your local shared drive at school. As it is already 160 lines it would be a slow and painful process to type it out manually!

AdventureGame Assignment.lua

<https://github.com/Inksaver/LuaForSchools/blob/main/Beginners/Section6/02-Adventure%20Game%20Assignment.lua>

Add a new folder called “lib” and copy/paste or download the kboard.lua library into it

kboard.lua

<https://github.com/Inksaver/LuaForSchools/blob/main/Beginners/Section6/lib/kboard.lua>

You will be using a more advanced type of table than you have used so far such as:

```
colours = {"red", "green", "blue"}
```

which is similar to a Python ‘List’ and can be accessed using a numerical index:

```
colour = colours[1]
```

(returns “red” as that is index 1).

WARNING all other languages use 0 based indexes, so in Python this would return “green”

The location tables use a text based index which is called a ‘key’.

This is the same as a Dictionary in other languages.

Example: the office table is created empty as before with

```
office = {}
```

Next you give it a name property

```
office.name = “office”
```

This is similar to a Python Dictionary and can be accessed with a string key:

```
office[“name”] = “office”
```

The dot notation and key are interchangeable: `office.name` is the same as `office[“name”]` but the dot notation is easier to understand.

<CENSORED>: Anybody below year 12 should NOT be reading this as it will cause neural malfunction and/or irreversible brain damage.

Those of you familiar with Object Oriented Programming (OOP) may be seeing familiar concepts. The tables are effectively static classes.

`office.name` is a public string property of the office class.

You could embed functions inside the office class. These would be static methods.

Example: in the kboard.lua library the method `getString()` is used to return a string of length `minInt` to `maxInt`, boolean `withTitle` for Title Case:

```
function Kboard.getString(prompt, withTitle, minInt, maxInt)
```

This is a method of the static class kboard.

</CENSORED>

General Instructions:

1. Import the kboard.lua library using require. If it is inside another folder eg 'lib' use 'lib.kboard' instead.
2. Plan your adventure and create some suitable locations using appropriately named tables eg 'office'.
3. Create the properties of each location with the same properties in each eg office.name, office.description
4. Make sure you include all four directions eg office.north, office.south
5. Use a table for items eg office.items = {} or office.items = {"paper", "pen"}
6. This is an example of a 'nested' list-type table inside the 'office' table

Instructions for main()

You need to construct a Game Loop. As it stands the function play(locationName) takes the current location and gets the user to select a new location (or 'Quit') which is returned to the variable locationName.

<clue>

Use either a while do loop or repeat until loop to continuously call the play() function.

If the user chooses to quit then the variable locationName will contain the string "Quit"

</clue>

Instructions for intro()

Change the text to make a suitable introduction for your adventure. You can use utf8 box characters by copy/pasting them from the top of the file.

The screen output will look exactly as it appears in your code.

You can add additional print statements if you wish

Instructions for setupPlayer()

The player table is given a default set of properties such as name, health, strength.

Add, edit or delete these properties to suit your adventure.

Change the list: characters = {"Genius", "Average Joe", "Teacher", "Earthworm"} to suitable names for your adventure.

Change the list of weapons.

Maybe ask further questions and use if statements to modify the properties further

Add some properties of your own.

Instructions for displayPlayer()

Finish printing out the remaining properties as neatly as possible

If you have given the player some items in player.inventory use a for loop to add them to the list based on the length of the list.

The # symbol can be used here where it means 'lengthOf'

```
<clue>
if #player.inventory > 0 then
  for index = 1, #player.inventory do
</clue>
```

Instructions for play()

This is the central function. It is called repeatedly by the game loop and displays information about the current location, any items in the area, and a choice of directions to take.

At this stage no options to interact with items has been given

A for loop is needed to display any items found in location.items based on the number of items.
(see clue above)

Instructions for getExits()

A new table is created using table.insert(<table>, <value>). It has been partially completed.
Add 2 more if statements to add east and west to it.