# Project Profile

This project requires you to implement a [text adventure game](https://en.wikipedia.org/wiki/Adventure_game) using the C programming language. A text adventure game is a form of [interactive fiction](https://en.wikipedia.org/wiki/Interactive_fiction) in which players use text commands to control characters, influence the environment, and navigate a virtual world. The first text adventure game developed between 1975-1977 was called [Colossal Cave Adventure](https://en.wikipedia.org/wiki/Colossal_Cave_Adventure). If you have never played a text adventure game you should [give it a try](https://www.amc.com/shows/halt-and-catch-fire/exclusives/colossal-cave-adventure) to get the idea. The goal of this project is to practice various aspects of the C programming language to help you prepare for implementing a text adventure of your own.

## Game State

At the heart of a text adventure game is the state of the world that you interact with. This can include players, rooms, items or objects, and anything else that one could imagine simulating in a virtual environment. The virtual world consists of a set of “rooms” that are linked together to form locations that can be navigated through directions such as **north**, **south**, **east**, **west, up** and **down**. A room has a description as well as objects that can be **taken** by a player, or more typically referred to as an [avatar](https://en.wikipedia.org/wiki/Avatar_(computing)), controlled by the user of the game. An avatar can be a character that is controlled by a user or it can be a [non-player character](https://en.wikipedia.org/wiki/Non-player_character). You could also extend the notion of a player to one that is controlled by an AI. An avatar can also have an inventory (e.g., backpack) that can hold items collected as they navigate from room to room. Items can and should be **used** to change the state of a room (i.e., unlock a secret room or win the game).

## Game Play

A text adventure is played by placing a player’s avatar in a starting room or location and allowing the user to provide text input. To get an idea of how this works you can check out many [example plays](https://youtu.be/O3etkSoHrR8) of the adventure game online. In general, there are several actions that can be played in a particular location including navigation such as *“look”* to provide a description of the location the player is currently in or *“go west”* to navigate to a new location (if you are able to go that direction). You can also pick up objects such as *“take key”*, apply an object in a location with *“use key”*, or get rid of an object with *“drop key”*. The objective of a text adventure game is up to the designer of that game. It could be to successfully navigate out of a maze of twisty passages or find a hidden item in a location and deliver it to a final location. The possibilities are endless, however, these games often involve a puzzle the player must solve and very often contain an [easter egg](https://en.wikipedia.org/wiki/Adventure_(Atari_2600)#Easter_egg) to make it more interesting.

The goal of this project is to implement a functional text adventure game given the requirements below.

# Requirements

Your primary objective is to use the C programming language to design and implement a functional text adventure game as described in the previous section. You are required to design and implement the appropriate data structures and corresponding algorithms that will enable a human player navigate their avatar through a virtual world and solve a puzzle that you design. The general design of the game is up to you, however, you must satisfy the following requirements:

## Game Requirements

1. Your game must have at least 8 rooms.
2. Your game must allow rooms to contain items.
3. Your game must implement an avatar where the avatar has an inventory (a place to put items).
4. Your game must allow an avatar to “see” the room they are in with the command “look”.
5. Your game must allow an avatar to move through each room using the command “go DIRECTION” where DIRECTION is a direction: north, south, east, west, up, down.
6. Your game must allow an avatar to pick up items with the command “take ITEM” where ITEM is the name of an item in a room.
7. Your game must allow an avatar to use an item with the command “use ITEM” where ITEM is the name of an item in the avatars inventory. The effect of using an item depends on the room the avatar is in.
8. You must implement at least one instance where an item alters the state of a room that the avatar is in.
9. Your game must allow an avatar to drop an item with the command “drop ITEM” where ITEM is the name of an item in the avatars inventory.
10. Your game must have a winning state (allowing the game to end).

## Code Requirements

1. You must use C structs in your code to represent various game objects.
2. You must use pointers in your code.
3. You must use dynamic allocation in your code using **malloc**/**calloc**.
4. You must deallocate memory in your code using **free**.

You are encouraged to be as creative as you would like with this game. Your virtual world can be anything ranging from fantasy and sci-fi to places at home, or vacation. Your virtual world can be inside a building or outside, the possibilities are endless, but it is up to you to come up with the world. You are also welcome to introduce additional commands that allow your avatar to interact with the virtual world in various ways.

# Code Organization

You should organize your code into multiple C files. In particular, you are to have the following files:

* **rooms.c**, **rooms.h** - the source and header file implementing data structures and functions for rooms.
* **items.c**, **items.h** - the source and header file implementing data structures and functions for items.
* **adventure.c** - the source file with functions to read user input and interpret commands as well as the main function.
* **Makefile** - the **Makefile** to use with the **make** command to build your code.

# 

# Deliverables

You must submit the following:

* **adventure.c, rooms.c**, **rooms.h**, **items.c**, **items.h** - this is your implementation of the text adventure game.
* **Makefile** - the build file to use with the **make** program.
* **README.txt** - this is a text file (made in Vim, Emacs, or Nano - not Word, TextEdit, etc.) containing an overview/description of your work highlighting the important parts of your implementation. You should also explain where in your implementation your code satisfies each of the requirements of the project. The goal should make it easy and obvious for a person to find the important rubric items.