Python Project

Text Adventure Game

Overview: Write a Python program to allow users to play a simple text adventure game based on a configuration file that you read in.

Learning objectives: Gain experience processing text files, building data structures using Python lists and dictionaries, and processing user input.

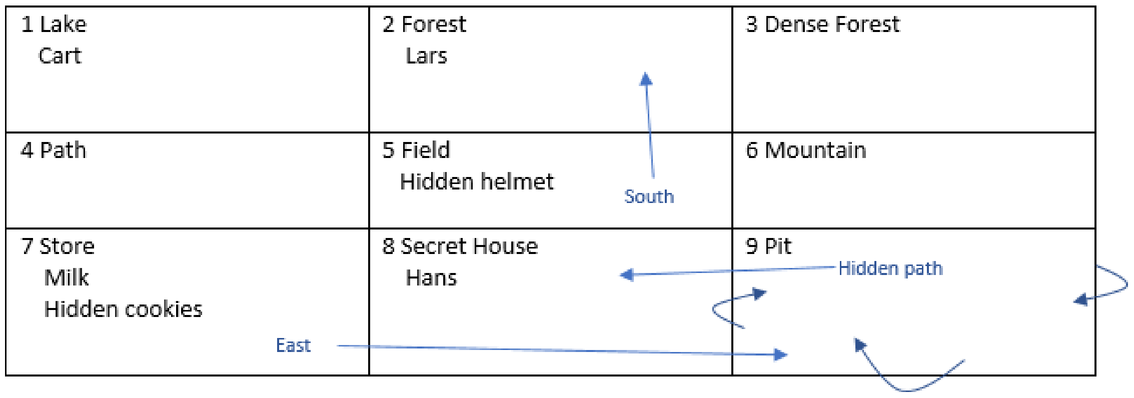
Project specification:
When personal computers were first entering the consumer market in the 1980s, their graphics capabilities were very limited. Text adventure games (such as Colossal Cave Adventure and Zork) were a popular form of maze+puzzle type of game. Below is a short example to illustrate how these games worked (text in bold was entered by the user).

```
You are standing next to a lake. The water is murky.
There is a cart here.
What next? take cart

You are standing next to a lake. The water is murky.
What next? inv

Your inventory: ['cart']
You are standing next to a lake. The water is murky.
What next? move west

You are in a very dense forest
What next? move west
```

These text adventure games often involved moving around a set of locations, finding objects, solving puzzles, and searching for items. "Winning" the game involved achieving some goal such as taking a specific object to a specific location or giving a specific object to a specific non-player character.

For this project, you will write a Python program that allows users to play a simple text adventure game based on a configuration file that you read in. An example configuration file and sample game run are shown at the end of this assignment.

Map and locations – For simplicity, the game will use a **rectangular** X by Y map of locations numbered from 1 to X*Y (e.g. 3x3 = 9 locations). Players can move around the map by issuing "move" commands followed by a direction (e.g., "move north"). Locations on the edges of the map should "wrap-around". For example, in the example map shown below, moving north from location 1 should move the player to location 7. Similarly, moving west from location 1 would take the player to location 3. The example map shown is square (3x3), but maps could be rectangular (X by Y).



Location descriptions – each location in the game has a brief description, which may contain hints.

Movement overrides – To make the game more interesting (and difficult!), locations may override the default movement behavior. For example, in the map above, if a player tries to move east, west, or south from location 9, they will end up back at location 9. However, moving north from location 9 would result in location 6. In location 5, all the movement is normal except for moving south, which will result in location 2 instead of location 8.

Hidden paths – To further add to the fun (and difficulty), locations may have hidden paths. Before a player can move along a hidden path, they must first discover it by issuing a "search" command in that location. After a player has discovered a hidden path, they can move along it by entering "move path". For example, location 9 has a hidden path that (once discovered) can move a player to location 8.

Objects and personal inventory – Objects in the game can be found at locations. When a player enters a location that contains an object, the game should print a notification that there is an object present (e.g., "There is a cart here."). When a player is in a location that contains an object, the player may take the object into their personal inventory using the "take" command followed by the name of the object (e.g. "take cart"). A take command removes the object from the location and places it into the player's inventory. A player can use the drop command to drop an object from their inventory and leave it in the current location (e.g., "drop cart"). Locations can contain any number of different objects. The player's inventory can also contain any number of different objects. Your code to support taking and dropping object should be very careful not to allow players to take objects that are not in the current location, or to drop objects that they are not carrying.

Hidden objects – To make the game even more fun and challenging, locations may have hidden objects. Before a player can "see" a hidden object, they must first discover it by issuing a "search" command in that location. After a player has discovered a hidden object, it should become visible just like any other object that is contained at that location.

Player interactions – Each time a player issues a command, the game should print a description of the current location and any objects that are visible.

Goal (win condition) – Each game will have a goal (winning condition) that involves placing a specific object at a specific location (e.g., "place the helmet at the location where Hans is"). This means that after every command a player issues, your program will need to check to see if the winning condition has been met (i.e., is the goal object at the goal location). When the winning condition is met, the game should print a congratulatory message and exit the game.

Player commands – Players can enter commands described below. Your program should handle all these commands and should not crash or exit unless the player issues the “exit” command or wins the game. Note: you can always press Control-C to interrupt a running Python program.

exit	Quit the game
inv	Show the player’s current inventory
goal	Show the goal of the game
search	Search for hidden objects and paths in the current location
take [object]	Remove the object from the current room and place it in the player’s current inventory.
drop [object]	Remove the object from the player’s inventory and place it in the current room.
move [direction]	Move in the direction specified (north, south, east, west)
move path	Move along a hidden path that has been found by searching

Configuration file

Your program should begin by reading in a game configuration file that contains information about the game, the goal, the locations, and objects. For simplicity, the game will use a rectangular grid of locations.

The configuration file should be formatted as shown in the example configuration file and should support the parameters described below.

• game_name:	Name of the game
• game_goal:	Description of the goal of the game
• game_goalloc:	The number of the goal location
• game_goalobj:	The object that must be placed at the goal location to win
• game_start:	The starting location number
• game_xsize:	The X dimension of the game rectangle (starting from 1)
• game_ysize:	The Y dimension of the game rectangle (starting from 1)
• ---	A line with only three dashes should be ignored
• r_id:	The number of the location that the following lines apply to
• r_desc:	The description of the location
• r_obj:	One object that is contained at the location
• r_north:	Override ‘north’ movement for this location to go to location num specified
• r_south:	Override ‘south’ movement for this location to go to location num specified
• r_east:	Override ‘east’ movement for this location to go to location num specified
• r_west:	Override ‘west’ movement for this location to go to location num specified
• r_hiddenobj:	One object that is hidden at the location
• r_hiddenpath:	Indicates that this location has a hidden path to the location number specified

I have posted two example configuration files on Sakai: `game1.txt` and `game2.txt`. Your Python program should be able to read either of these configuration files (or any other properly formatted configuration file). When downloading text files from Sakai, you should use your browser’s “Save as” feature to save the file exactly as it is posted rather than trying to cut and paste the text into a new file.

Data Structure Requirements

Your program **must** implement the following data structures to support the game play.

• inventory	a list of the items that the player is carrying
• gameinfo	a dictionary with the game name, goal, goalloc, start, xsize, and ysize
• map	a dictionary of location numbers which may include additional dictionaries and/or lists (see below)

An example of the **map** data structure that I built for the example configuration file is shown below:

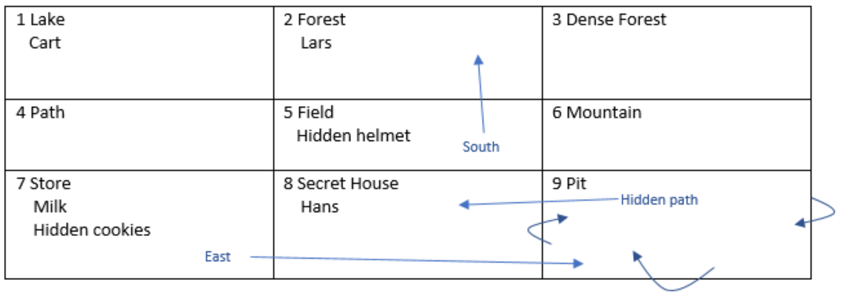
```
{
    "1": {
        "desc": "standing next to a lake. The water is murky.",
        "id": "1",
        "obj": [
            "cart"
        ]
    },
    "2": {
        "desc": "in a forest. Lars the hunter is here.",
        "id": "2",
        "obj": []
    },
    "3": {
        "desc": "in a very dense forest",
        "id": "3",
        "obj": []
    },
    "4": {
        "desc": "on a path",
        "id": "4",
        "obj": []
    },
    "5": {
        "desc": "in an open field.  It looks like there was a battle
here a long time ago.",
        "hiddenobj": "helmet",
        "id": "5",
        "obj": [],
        "south": "2"
    },
    "6": {
        "desc": "at the foot of a mountain",
        "id": "6",
        "obj": []
    },
    "7": {
        "desc": "inside a general store.",
        "east": "9",
        "hiddenobj": "cookies",
        "id": "7",
        "obj": [
            "milk"
        ]
    },
    "8": {
        "desc": "inside a secret house. Hans is here.",
        "id": "8",
        "obj": []
    },
    "9": {
        "desc": "in a pit!  You must have accidently fallen into it.",
        "east": "9",
        "hiddenpath": "8",
        "id": "9",
        "obj": [],
        "south": "9",
        "west": "9"
    }
}
```

Advanced functionality
If you implement the program as outlined so far, you can earn up to a maximum of 95 out of 100 points. You can earn additional points for implementing additional features:

(5 points) Implement a new command "talk" that a player can use to talk to a non-player character (NPC) who is at the same location (e.g., "talk Hans"). The configuration file will contain lines with the location of the NPC and two responses for each NPC – one that they give the first time the player "talks" to them and a second response that they give for all subsequent "talk" commands:

```
npc_Lars_loc:   2
npc_Lars_1:     Welcome traveler!
npc_Lars_2:     I heard that Hans has a secret house near the pit.
```

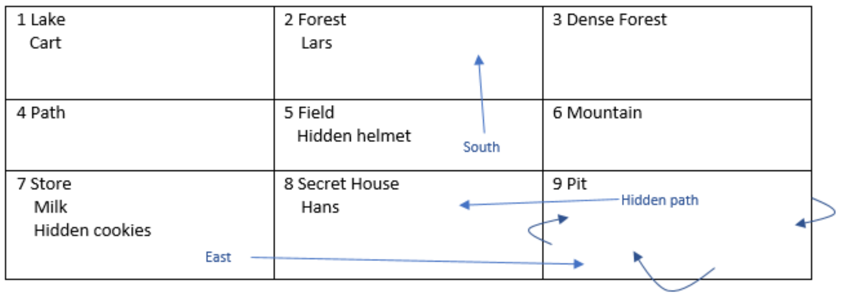
Game 1 Configuration File



```
game_name: Adventure 1
game_goal: Find the magic
helmet and bring it to Hans.
game_goalloc: 8
game_goalobj: helmet
game_start: 4
game_xsize: 3
game_ysize: 3
---
r_id:1
r_desc: standing next to a lake. The water is murky.
r_obj: cart
---
r_id:2
r_desc: in a forest. Lars the hunter is here.
---
r_id:3
r_desc: in a very dense forest
---
r_id:4
r_desc: on a path
---
r_id:5
r_desc: in an open field.  It looks like there was a battle here a long
time ago.
r_hiddenobj: helmet
r_south: 2
---
r_id:6
r_desc: at the foot of a mountain
---
r_id:7
r_desc: inside a general store.
r_obj: milk
r_hiddenobj: cookies
r_east: 9
---
r_id:8
r_desc: inside a secret house. Hans is here.
---
r_id:9
r_desc: in a pit!  You must have accidently fallen into it.
r_south: 9
r_east: 9
r_west: 9
r_hiddenpath: 8
```

Example Program Run

```
Welcome to Adventure 1

The goal of this game is to:
Find the magic helmet and bring it to Hans.

You are  on a path
What next? move north

You are  standing next to a lake. The water is murky.
There is a  cart here.
What next? take cart

You are  standing next to a lake. The water is murky.
What next? inv

Your inventory:  ['cart']
You are  standing next to a lake. The water is murky.
What next? move north

You are  inside a general store.
There is a  milk here.
What next? take milk

You are  inside a general store.
What next? inv

Your inventory:  ['cart', 'milk']
You are  inside a general store.
What next? drop cart

You are  inside a general store.
There is a  cart here.
What next? inv

Your inventory:  ['milk']
You are  inside a general store.
There is a  cart here.
What next? move north

You are  on a path
What next? move east

You are  in an open field.  It looks like there was a battle here a long time ago.
What next? take helmet

That object is not here.
You are  in an open field.  It looks like there was a battle here a long time ago.
What next? search

You found a  helmet
You are  in an open field.  It looks like there was a battle here a long time ago.
There is a  helmet here.
What next? take helmet

You are  in an open field.  It looks like there was a battle here a long time ago.
What next? inv

Your inventory:  ['milk', 'helmet']
You are  in an open field.  It looks like there was a battle here a long time ago.
What next? move south

You are  in a forest. Lars the hunter is here.
What next? move east

You are  in a very dense forest
What next? move north

You are  in a pit!  You must have accidently fallen into it.
What next? move west

You are  in a pit!  You must have accidently fallen into it.
What next? move east

You are  in a pit!  You must have accidently fallen into it.
What next? search

You found a hidden path!
```



```
You are  in a pit!  You must have accidently fallen into it.
What next? move path

You are  inside a secret house. Hans is here.
What next? goal

The goal of this game is to:
Find the magic helmet and bring it to Hans.
You are  inside a secret house. Hans is here.
What next? drop helmet

Congratulations!  You have won the game.
>>>
```