By submitting this assignment, I agree to the following:

"Aggies do not lie, cheat, or steal, or tolerate those who do."

"I have not given or received any unauthorized aid on this assignment."

Names: Lindsey Wilkin, Nishi Mishra, Charlie Simpson, and Kai Brown

Section: 543

Assignment: Project User Manual

Date: November 25, 2020

Valid Inputs

Function Name	**Input type**	**Input Example**
addGold_to_player(gold_amount)	integer	addGold_to_player(100)
removeGold_from_player(gold_amount)	integer	removeGold_from_player(10 0)
addXP_to_player(num_XP)	integer	addXP_to_player(25)
prompt(*question)	variable amount of strings	prompt(“You want (1) or (2)”, “1”, “2”)
prompt2(question,answers)	String, list of strings	“Question Text”,”Answer Text” **
get_inventory_value(action) #looks at inventory only	string	get_inventory_value(“Which item would you like to sell? \n(type e to exit)”
get_inventory_value2(action) #looks at inventory and equipped items	string	get_inventory_value2("Which item would you like to inspect? ")
town(x)	integer	town(0) 0 or 1 #0 = first time #1 = not first time

Valid Outputs

Function Name	Output type	Output Example
addGold_to_player(gold_amount)	none	data[1] is updated
removeGold_from_player(gold_amount)	none	data[1] is updated
addXP_to_player(num_XP)	none	data[3] is updated updatelvl() is called
prompt(*question)	A 'correct' answer from the list given by function	prompt("You want (1) or (2)") output: "2"
prompt2(question,answers)	A 'correct' answer from the list given by function	This function was made specifically for the shop() func
get_inventory_value(action)	The index of the item in inventory as an int	--IN HAND-- Bronze Sword Output: 2
get_inventory_value2(action) #identical functions, they just search through different amounts of items	The index of the item in inventory as an int	--IN HAND-- Bronze Sword --INVENTORY-- Leather Sheaves Output: 3
town(x)	Gives Menu Options as a string	Welcome to the Town Square...

Description of Code

Our project is a text adventure. Its structure is similar to that of Super Mario Brothers. Both games involve gaining loot which puts the user in a better position to advance through the levels. In our program the user gains loot to buy better armor and weapons.

The basic idea for our game is the user will have access to a shop to buy weapons, armor, and healing to protect themselves while in the dungeon. In the shop the user can buy these items using the loot they have obtained throughout the game. When in the dungeon the

user will encounter several events. These events include loot drops, sayings, and monster fights. These events will be randomized so the user will encounter multiple of each event in a single level.

The loot drops are when the user can get more money to buy items from the shop. Sayings are strings of words printed to the console that are used to immerse the user into the game. Finally, the monster battles are when the user is faced with an opponent and is promoted to input a combination of attacks and/or defenses until the opponent's health is depleted.

There are a total of 20 levels in this game. After each level, the user will be able to buy new items from the shop to use in the following levels. In addition, the monster fights will progressively get harder each level.

The goal of this game is to collect loot, upgrade your armor, and fight off monsters until you have completed level 20.

Description of Specific Pieces of Code...

Items Dictionary: Holds all of the possible items that can be bought from the store

Levels list: Lists level number and the number of events to happen in each level

monster(): Defines the current monster's name, health, defense, XP gain, and gold gain

data list: Defines the player's inventory (equipped weapon & armor, other bought items) and holds the values for the player's current gold amount, current level, current XP amount, current health, baseline health (aka maximum health they can currently obtain) and level completion status. (Array of 20 zeros, each index represents that game level number. 0 == not completed, 1 == completed)

saying(): Creates a list of roleplaying phrases and returns one randomly from the list

addGold_to_player(gold_amount): Adds whatever amount of gold imputed into the function

removeGold_from_player(gold_amount): Removes gold amount passed into function

update_lvl(): Function checks if player is able to level up. If they can, then their level is updated and the user's health is increased by 10. These do direct changes to the data array to update values to prepare for the next level

addXP_to_player(num_XP): Function adds XP to player and checks if the player is able to increase their level after their XP increase. data[3] refers to player XP and update_lvl() is called inside

save(): Saves the game progress by writing to file

prompt(*question): This function checks if the input is valid. You would call it with: prompt("Question text", "Possible answer 1", "Possible answer 2", keep going for as many possible answers exist.) Returns the answer the user selected.

prompt2(question,answers): Use this version if the number of potential answers can vary.

Syntax: prompt("Question text", [all possible answers in this specific instance,2,3,4,5,etc])

Returns the answer the user selected.

shop(): The shop is where the user will come to buy and sell items in their inventory

viewinventory(): This function prints every item in the inventory and is called in the checkInventory() function.

get_inventory_value(action): Allows the user to look through specific items in their inventory. This function does NOT include the user's equipped items.

get_inventory_value2(action): Allows the user to look through specific items in their inventory. This function does include the user's equipped items.

checkInventory(): This function allows/prompts the player to inspect or swap items in the inventory. This function calls the swap() and inspect() functions as needed. This mostly presents the menu and navigation options.

inspect(index): This function gets the intex in the items array and displays the property of that item via a print statement

swap(item_in, item_out): Takes in two indexes, the first one refers to the item being swapped in and the other swapped out. This function allows the user to equip different weapons/armor. This directly changes the data array containing the current armor and weapon equipped.

town(x): The town function welcomes the user and provides options of how to proceed into the game. This function is for organization purposes.

getLoot(): This function will be one of the three functions called in the next event function. If called, the user will gain a randomized amount of loot in the dungeon.

battle_monster(): This function determines the monster's attacks and defense for the current level.

battle_user(): This function determines the users attacks for the current level.

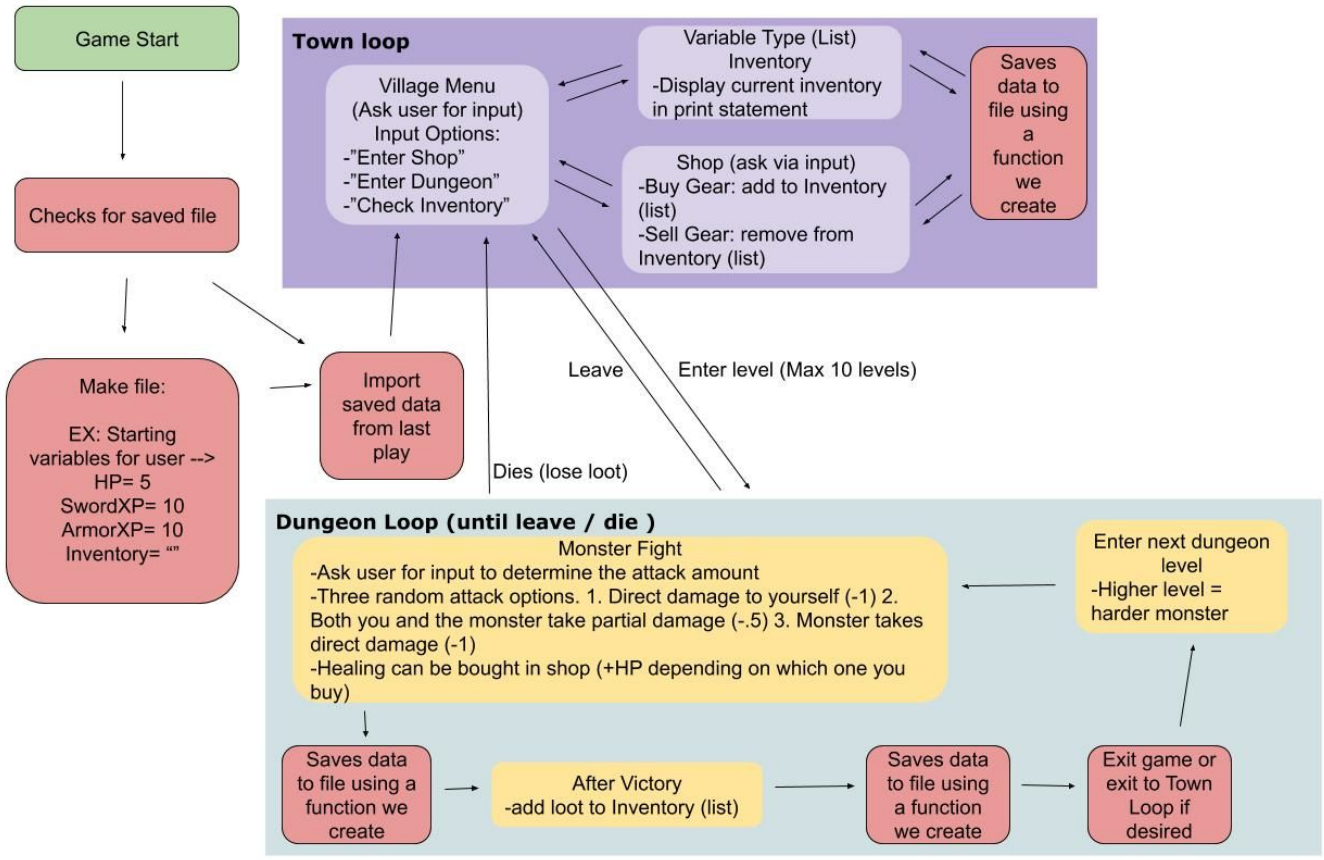
battle(): This function runs the monster battles. The code matches up all of the possible outcomes of monster/user attacks and defenses and prints a statement to the user. The math behind monster/user attacks and defenses are also calculated in this code.

dungeon(): This function is the main function of the level. It runs every event.

nextEvent(): This function decides the next event the user will encounter while in the dungeon.

***** Additionally, there are seven functions within this program that are used to print the text in color. These functions use ANSI escape codes to do so, and are our “learn something new”. *****

Flowchart of Project



Game Walk-Through

1) You will be prompted to change your settings on Spyder so your background is the darkest black. This ensures colors will show up properly.

```
On Spyder for the colors to properly show please change your
background to the darkest black.

To change:
Tools > Preferences > Appearance > Syntax Highlighting Scheme >
Create New Scheme > Background (bottom right -- click the color
box) > move the color slider to the darkest black
```

2) A welcome screen containing the logo and goal of the program will be displayed along with a choice to open a saved game or a new game.

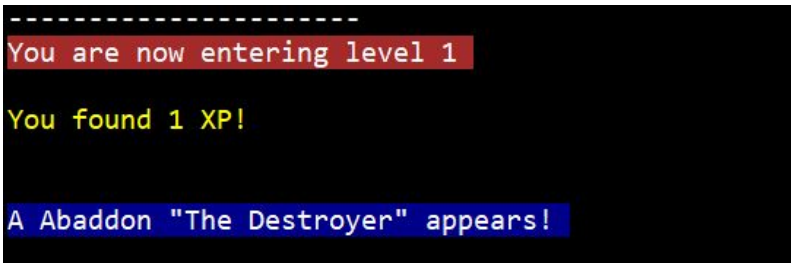
```
----------------------
|                    |
|  UWU KnightsQuest  |
|                    |
----------------------


This program's goal is to provide entertainment by immersing the
user into a text adventure
You will be prompted to provide choices throughout the game. Follow
the prompts.

Open saved game (o) | Start new game (n): n

----------------------
```

3) Choosing “n” you can continue pressing enter following the prompts. Arriving in the town square you will need to choose to either play the game, visit shop, check inventory or save/quit.

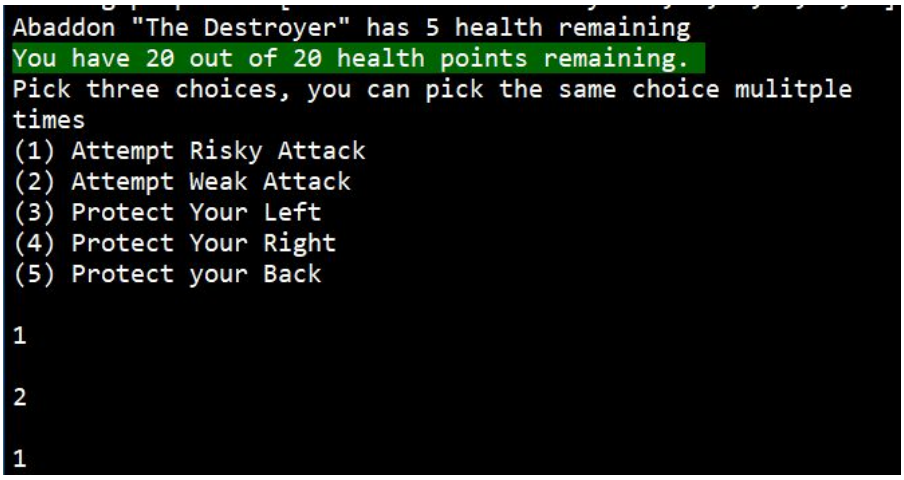
```
*You awake in your hometown.* (press enter to continue)

*Today you will become an adventurer.* (press enter to continue)
----------------------
Welcome to the town square!

Venture forth!  (1)
Visit Shop      (2)
Check inventory (3)
Save & Quit     (4)
Enter your decision: 1

----------------------
```

4) Upon selecting choice 1: venture forth! You will enter the first level. This will run the nextEvent() loop which chooses three potential events: finding XP/gold, printing a saying or fighting a monster. In this case, you found some loot and were given a saying.

5) In another instance of running the program, you are presented with some loot and a monster to defeat.



6) Upon hitting enter, you will be presented with 5 potential attack types. The same attacks can be reused. The risky attack is the most powerful, but could potentially harm yourself. The weak attack can only potentially harm the monster with no damage to you.



7) In response to your attacks or protections the monster's choices will be displayed. The monster's attacks are random with a percent weightage. It displays how much damage

was caused to both parties.

```
The Abaddon "The Destroyer" out-smarted you with an attack to the
back! You took damage!
monster attack (1) - armor protection (0) = total damage (1)


The risky attack paid off! Critical damage!
Weapon damage X 2 (2) - Monster armor / 2 (0) = net attack (2)


The risky attack paid off! Critical damage!
Weapon damage X 2 (2) - Monster armor / 2 (0) = net attack (4)
```

8) At the end of the round it will display damage and award gold/XP. (Typo is fixed & colors may have changed) However this screen is the beating monster screen.

```
total damage to player: 1
total damage to monstee: 5
You have defeated the monster!

You earned 1 gold and 1 xp!
```

```
Total damage to player: 2
Total damage to monster: 6
You have defeated the monster!
You earned 1 gold and 1 xp!
Enter darkess

You beat the level! You earned 1 Gold and 1 XP!
```

9) After each level (I played two) you will have the option of continuing or the other options.

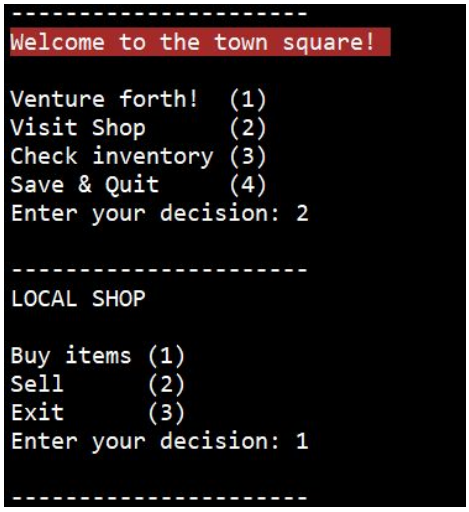
```
----------------------
Welcome to the town square!

Venture forth!  (1)
Visit Shop      (2)
Check inventory (3)
Save & Quit     (4)
Enter your decision: 4
----------------------
Game saved and closed.
```

```
'Butter Knife', 'T-shirt and Shorts'
4
1
3
13
20
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

The save function saves the function to file. The file is displayed on the right. The first line is inventory. It says I have 4 gold, my player is level 1. I have 3 XP. 13 health

currently. 20 is the player's baseline health (increases when the player levels up). The levels array shows I have completed two levels so far.

10) The town square menu also gives you three other options to choose besides going into a level.



a) If you choose to visit the shop, you will be able to buy and sell items.

b) If you choose to check your inventory, all of the items that are currently in your possession will be printed to the screen.

c) Lastly, if you choose save and quit, your data will be saved as shown above in step 8. This data can be reopened for the next time you play.

11) If you choose to buy items in the shop, you will be presented with a list printed to the console of all the items available to buy at your current level. To buy the item you will enter its number and then confirm you want to process the transaction. If the sale goes through you will receive a message saying "Item bought successfully!", the loot used for the purchase will be taken, and the item will be added to your inventory. Two other things could occur. Your inventory could be full, or you might not have enough loot to buy an item. If your inventory is full, you will see the message "Your inventory is full!". If you don't have enough loot, you will see "Aye, you no money! Get out of my shop!".

12) If your inventory is full, or you just want to get rid of something, you can choose to sell some of your items. To do so, enter a "2" in the local shop. This will again print a list of the item in your inventory, each with a key number. To sell the item, enter its key number and confirm the following "yes or no". Once confirmed, a message saying "Item sold successfully!" will be printed to your screen, you will receive loot for your sale, and the item will be removed from your inventory.

13) Once you have completed your time in the shop, you can enter a "3" to exit. This will bring you back out to the town square where you will continue the game.

14) All of the previous steps can and will be repeated until the user completes level 20. Once level 20 is completed, the user will have beaten the game and the game will end.