COM S 1270 - 'Candy Realm!'

Please see the assignment information on Canvas for this assignment's initial due date and final deadline.

Assignment Objective

The purpose of this assignment is to give you practice creating your own unique version of a pre-existing boardgame, 'Candy Land,' which will be in a file called `candyRealm.py`.

This project will likely require using all of the skills and techniques learned in the class thus far.

Instructions

You will program the game of 'Candy Realm!' This will be a game very similar to the classic boardgame 'Candy Land' by Hasbro. While some people have not had the opportunity to play the original game ('Candy Land'), this is a good opportunity to see what it is (it is not very complicated).

Your submission should generally follow the rules of the original game (found below), but it does not need to follow those rules precisely. Feel free to use your imagination a bit to make the game a bit more fun!

You may accomplish this assignment however you like outside of using ChatGPT/ AI code generation/ cheating. Please note that the instructor is already aware of most (if not all) of the 'Candy Land' Python tutorials on the first page of a Google search, as well as the output of ChatGPT for the same task. As a fair forewarning, this is the assignment that many cheat-cheaty-cheaters get caught on.

Your submission should visualize the game in some way. Meaning, that the user should be able to see the pieces/ board. The example version allows the user(s) to see the order of the card deck. However, this is not required and you may do things as you like.

Your submission does *not* have to exactly match the original game, but it should approximate the general idea. Meaning, a submission could have 'wildcards,' 'extra rules,' 'double drawing cards,' and basically anything else! You can feel free to use your imagination!

Your submission should have a 'main menu' where the user can ask for instructions for the game, play the game, and quit the game. Once the game is over, the program should return to this 'main menu' and reset all the relevant gameplay-related data.

Your submission should have the ability for the human to play the game against up to three AI 'agents' (which can be programmed any way you see fit), or three other humans, or any combination thereof (e.g. 1 human vs. 3 computers, 2 humans vs. 2 computers, 3 humans vs. 1 computer, or just 4 humans). In general, there should be a total of four players in the game.

As a word of warning/ advice - make sure that the players cannot get stuck in some sort of 'infinite loop' and can actually finish the game. The last space of the game can be a specific 'color' or it can take *any* 'color' - whatever you so choose.

Exmaple Rules:

- https://www.geekyhobbies.com/candy-land-board-game-rules-and-instructions-for-how-to-play/
- https://instructions.hasbro.com/en-us/instruction/Candy-Land-Game

- https://www.wikihow.com/Play-Candy-Land
- https://www.youtube.com/watch?v=Z4ZKPU-_6G0

Special Features:

If you would like, you can add colors to your output by way of the `colorama` module:

- https://pypi.org/project/colorama/

However, this is not required.

You can install colorama with the following commands:

- (PC): `pip install colorama`
- (Mac): `pip3 install colorama`

Other Requirements:

Additionally, you *do* need to compensate for the crashes that occur when the user enters a value of the incorrect type. (E.g., The program asks for an integer, but the user enters a letter instead.) This requirement is different from many of the other assignments, and will be one that will be checked for when grading your game.

It is required for you to write your name, the date you started working on your script, and a short explanation of what your code does at the top of your file. For example:

```
# Matthew Holman             7-19-2024
# Assignment #6
#
# This is a version of the classic boardgame 'Candy Land' - here called 'Candy Realm!'
```

Do *not* attempt to 'double dip' by placing your 'header' printout information at the top of your file - the header and the initial comment block are different things.

Your work should be your own, completely original effort. Meaning - you should not just copy my code explanation above, but rather you should try to come up with one of your own.

The actual Python script itself can be programmed in any way (excluding cheating ala ChatGPT) so long as the output resembles that below. This includes the use of the initial 'header' when starting the program (i.e. the program title, who it is by, the student's class/ section, etc.). Please note that the 'header' will be strictly enforced with this assignment. Be sure to include the title, your name, and your class/ section.

Example Output

```
Candy Realm!

By: Matthew Holman
[COM S 127 1]

-----------------------------------------------------------------
MAIN MENU: [p]lay game, [i]nstructions, or [q]uit?: p

How Many Human Players [1] - [4]?: 1
```

```
How Many Copies Of Each Card [1] - [5]?: 3

        1
        2
        3
        4
 START  M  R  B  M  C  G  B  Y  C  B  G  Y  G  C  Y  R  M  R  GOAL!
 CARDS  M  Y  M  B  G  R  G  R  Y  B  Y  C  C  C  B  G  M  R

Player 1: Would you like to [d]raw a  M  card, [s]huffle the deck, or [q]uit?: s

Player 1 (HUMAN) has shuffled the deck!
Press [ENTER] To Continue...

        1
        2
        3
        4
 START  M  R  B  M  C  G  B  Y  C  B  G  Y  G  C  Y  R  M  R  GOAL!
 CARDS  M  C  Y  R  M  Y  B  B  G  B  C  G  R  G  C  M  Y  R
Player 2 (COMPUTER) has shuffled the deck!
Press [ENTER] To Continue...

        1
        2
        3
        4
 START  M  R  B  M  C  G  B  Y  C  B  G  Y  G  C  Y  R  M  R  GOAL!
 CARDS  G  B  R  C  R  M  Y  M  B  Y  B  C  M  G  C  G  Y  R
Player 3 (COMPUTER) has drawn:  G
Player 3 (COMPUTER) moves forward 5 spaces!
Press [ENTER] To Continue...

        1
        2
                  3
        4
 START  M  R  B  M  C  G  B  Y  C  B  G  Y  G  C  Y  R  M  R  GOAL!
 CARDS  B  R  C  R  M  Y  M  B  Y  B  C  M  G  C  G  Y  R  G
Player 4 (COMPUTER) has shuffled the deck!
Press [ENTER] To Continue...

        1
        2
                  3
        4
 START  M  R  B  M  C  G  B  Y  C  B  G  Y  G  C  Y  R  M  R  GOAL!
 CARDS  B  R  B  R  M  R  Y  Y  Y  G  C  G  M  B  M  C  G  C

Player 1: Would you like to [d]raw a  B  card, [s]huffle the deck, or [q]uit?: d

Player 1 (HUMAN) has drawn:  B
Player 1 (HUMAN) moves forward 2 spaces!
Press [ENTER] To Continue...

            1
        2
                  3
        4
 START  M  R  B  M  C  G  B  Y  C  B  G  Y  G  C  Y  R  M  R  GOAL!
 CARDS  R  B  R  M  R  Y  Y  Y  G  C  G  M  B  M  C  G  C  B
Player 2 (COMPUTER) has drawn:  R
```

```
Player 2 (COMPUTER) moves forward 1 spaces!
Press [ENTER] To Continue...
```

etc.

Special Notes

NOTE: This assignment will the most difficult assignment of the semester. However:

- Completing this assignment may require you to start your work 'before the last minute.' Please plan accordingly.
- Your script **CANNOT** crash under any circumstances. Any portions of your code where the script crashes will receive a zero (0) for that aspect of the assignment.
    - Understand, when the word 'crash' is used in the instruction above, what this means is 'crashing under expected use cases given the level of knowledge attained in the class.'
        - You finally have the ability to enforce user input types. For example, if the script asks for an integer, and the user enters a letter, you can program the script in such a way that it no longer crashes in instances like this. Handling exceptions is now the expectation for all user input.

NOTE: You are turning in your CODE - NOT the OUTPUT of your code.

- Your code will be run by the TAs, and the output of those runs, along with the code itself, is what will be evaluated.

NOTE: Assignments turned in in any other format other the specified file types will not be accepted.

- Screenshots of code **will not** be accepted.
- `.sln` files are **not** code files - they contain **no** Python code and **will not** be accepted.
- `.zip`, `.rar`, `.tar.gz`, and other compressed files **will not** be accepted unless otherwise specified.
- If your submission is not in a `.py` file, when so specified, or if the submission is not accompanied by **all** the files needed to run the submission, the submission will not be graded.
    - **THIS WILL LEAD TO YOU RECEIVING A ZERO (0) ON THE ASSIGNMENT.**
    - **You will *NOT* be allowed to re-submit your work after the final deadline in this case.**

NOTE: You have the ability (and responsibility) to 'double check' that your work/ submission is correct when you turn it in.

- If you accidentally turn in the wrong submission, you will **not** be allowed to resubmit it after the final deadline.
- Please note, you can submit your work as many times as you like *before* the final deadline.