CPSC 231: Assignment 2

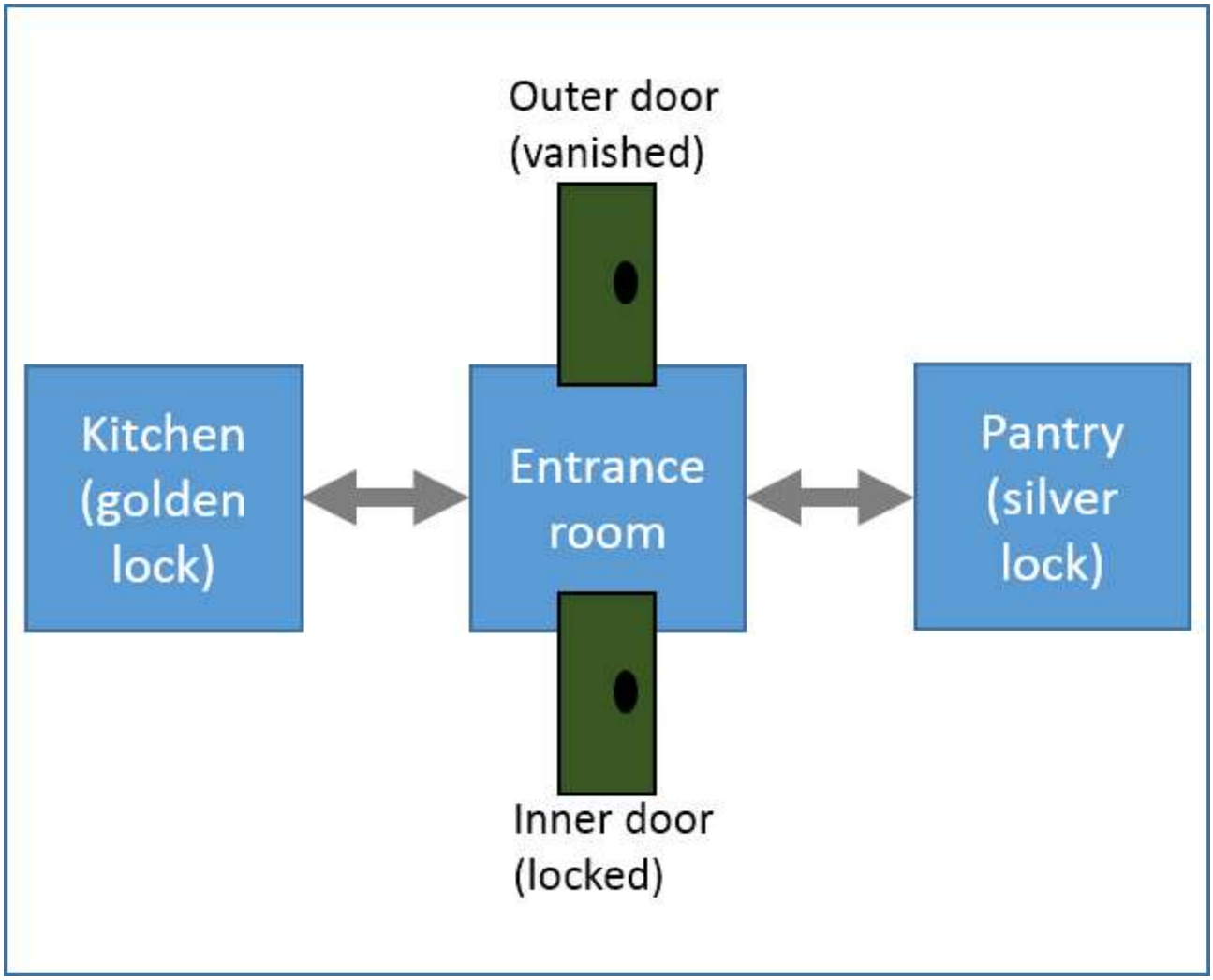
Due Friday June 2 at 5 PM

New Concepts to be applied for the assignment

- Problem solving using branches and loops in Python

Description:

Write a text-based "choose your own" adventure game. There will be three 'rooms' in the game world, within each room the player will see a number of choices and the program will react to the player's selection. The game will repeatedly run the game via a main loop (contains most of the instructions of the program) until the game has been won. The goal of this mini-game is to open the locked door to access the rest of the house. **The door will only be unlocked if the player turns the key silver lock in the pantry to the 'right' position and turns the key in the gold lock in the kitchen to the 'left' position.** When the door has been unlocked the player can return to the entrance room and open the door to the rest of the house satisfying the goal of this mini-game: the main loop ends and a suitable congratulatory message should be displayed.



Sample program output:

A text file "`output_normal_play.txt`" shows sample execution of my solution with a normal play through (no erroneous selections). Another text file "`output_invalid_selections.txt`" illustrates what should happen when the player selects an option that isn't listed. These files can be found in UNIX under `/home/231/assignments/assignment2/output`. Alternatively you can see the contents of the A2 directory via a [shortcut link].

Entrance room description

The player begins the game in the entrance. The door that brought the person into the house is gone (it's a "one-way portal"). When the player is in this room the game will describe the list of possible actions:

1. Try to open the door
2. Go through the left entryway
3. Go through the right entryway

As mentioned the player cannot open the door until it as been unlocked. The player can freely pass through either entryway with no conditions. Entering a non-existent selection will result in a suitable helpful error message (refer to the last section of the 'repetition' notes, rules of thumb for how to write good error messages) and the above options are displayed again. The program does not have to handle cases where an invalid type of information (e.g. string instead of number) has been entered. (However the inability to handle invalid types should then be documented as a "program limitation"). As long as the player remains in the room the program will repeatedly display the above menu options (using a loop) after each time that the user enters his/her selection.

Pantry description

As shown in the above map the player can freely pass between the entrance and the pantry. When the player is in the pantry, the game will display the list of possible actions:

1. Turn the silver lock to the left position
2. Turn the silver lock to the right position
3. Turn the silver lock to the center position
4. Don't change the position! Return to entranceway

The program must show the current position of the lock e.g. "The silver lock is currently set to the right position.". The starting default is the 'left' position. Similar to the entrance if the player enters an invalid selection in the pantry, a suitable and helpful error message should be displayed and the list of options for the pantry will be shown again. As long as the player remains in the room the program will repeatedly display the above menu options (using a loop) after each time that the user enters his/her selection.

Kitchen description

As shown in the above map the player can freely pass between the entrance and the kitchen. When the player is in the kitchen, the game will display the list of possible actions:

1. Turn the gold lock to the left position
2. Turn the gold lock to the right position
3. Turn the gold lock to the center position
4. Don't change the position! Return to entranceway

The program should show the current lever position e.g. "The gold lock is currently set to the center position.". The starting default is the 'center' position. Similar to the entrance if the player enters an invalid selection in the kitchen, a suitable and helpful error message should be displayed and then the list of options for the kitchen will be shown again. As long as the player remains in the room the program will repeatedly display the above menu options (using a loop) after each time that the user enters his/her selection.

In addition to grading on whether the above functionality was correctly implemented TAs will also look at style and documentation.

Documentation (synopsis from the intro to programming notes)

- Contact information (name, student ID, tutorial number)
- Header documentation: summarize the functionality of program
- Inline documentation: for each room describe the specific features from the assignment description that were implemented for the room
- Program limitations (if applicable)

Style (synopsis from the intro to programming notes)

- Good naming conventions
- The use of named constants as appropriate
- Clear expressions (mathematical, Boolean)
- Appropriate use of whitespace (not too much or too little)

D2L configuration:

- Multiple submissions are possible for each assignment: You can and should submit many times before the due date. D2L will simply overwrite previous submissions with newer ones.
- **Important**! Multiple files can be submitted for each assignment. I am allowing you to submit multiple files for each assignment so **don't archive/compress multiple files using a utility such as zip**. However, this means that **TAs will only mark the latest versions** of each file submitted via D2L. Even if the version of a document that you want marked has been uploaded into D2L if it isn't the latest version then you will only get marks for the latest version. (It's unfair to have the TAs check versions or to remark assignments because marking is enough work as-is).

Marking

- Assignments will be marked by your tutorial instructor (the "Teaching Assistant" or "TA"). When you have questions about marking this is the first person that you should be directing your questions towards. If you still have question after you have talked to your TA, then you can talk to your course (lecture) instructor.
- As well as being marked on whether "your program works" you will also be marked on non-functional requirements such as style and documentation. Consequently this assignment will include a separate [marking checklist]. Besides seeing your grade point in D2L you can also see the detailed feedback that your TA will enter for each student. You can access the grading sheet in D2L under `Assessments->Dropbox` and then clicking on the appropriate assignment link. If you still cannot find the grading sheet then here is a [help link]

Points to keep in mind:

1. **Due time:** All assignments are due at 5 **PM** on the due dates listed on the course web page. Late assignments or components of assignments will not be accepted for marking without approval for an

extension beforehand. Alternate submission mechanisms (non exhaustive list of examples: email, uploads to cloud-based systems such as Google drive, time-stamps, TA memories) cannot be used as alternatives if you have forgotten to submit work or otherwise have not properly submitted into D2L. **Only files submitted into D2L by the due date** is **what will be marked**, everything else will be awarded no credit.

2. **Extensions** may be granted for reasonable cases by the course instructor with the receipt of the appropriate documentation (e.g., a doctor's note). Typical examples of reasonable cases for an extension include: illness or a death in the family. Example cases where extensions will not be granted include situations that are typical of student life: having multiple due dates, work commitments etc. Tutorial instructors (TA's) will not be able to provide extension on their own and must receive permission from the course instructor first.

3. **Method of submission:** You are to submit your assignment using D2L [help link]. Make sure that you [check the contents of your submitted files] (e.g., is the file okay or was it corrupted, is it the correct version etc.). It's your responsibility to do this! (Make sure that you submit your assignment with enough time before it comes due for you to do a check).

4. **Identifying information**: All assignments should include contact information (full name, student ID number and tutorial section) at the very top of your program in the class where the '`main()`' method resides (starting execution point). (Note other documentation is also required for most assignments).

5. **Collaboration**: Assignments must reflect individual work; group work is not allowed in this class nor can you copy the work of others. For more detailed information as to what constitutes academic misconduct (i.e., cheating) for this course please read the following [link].

6. **Execution**: programs must run on the computer science network running Python 3.x. If you write you code in the lab and work remotely using a remote login program such as Putty or SSH. If you choose to install Python on your own computer then it is your responsibility to ensure that your program will run properly here. It's not recommended that you use an IDE for writing your programs but if you use one then make sure that you submit your program in the form of text "`.py`" file or files.

7. **Use of pre-created Python libraries**: unless otherwise told you are to write the code yourself and not use any pre-created functions. For this assignment the usual acceptable functions include: `print()`, `input()` and the 'conversion' functions such as `int()`, `float()`, `str()`. Look at the particular assignment description for a list of other classes that you are allowed to use and still get credit in an assignment submission.