CPSC 231: Assignment 3

Due Monday June 12 at 4 PM

New Concepts to be applied for the assignment

- Decomposing a problem into functions (includes concepts such as parameters, return values, local variables/scope)

Description:

Implement the functionality for the previous assignment using functions. One way of decomposing your program is to implement the process of instructions for a room as a function or functions. Because there are six rooms that will mean that your program will consist of at least 7 functions (1 per room plus a starting function). You may be able to subdivide your program using an alternate approach to writing 1 function per room but your program must follow good design principles for using functions.

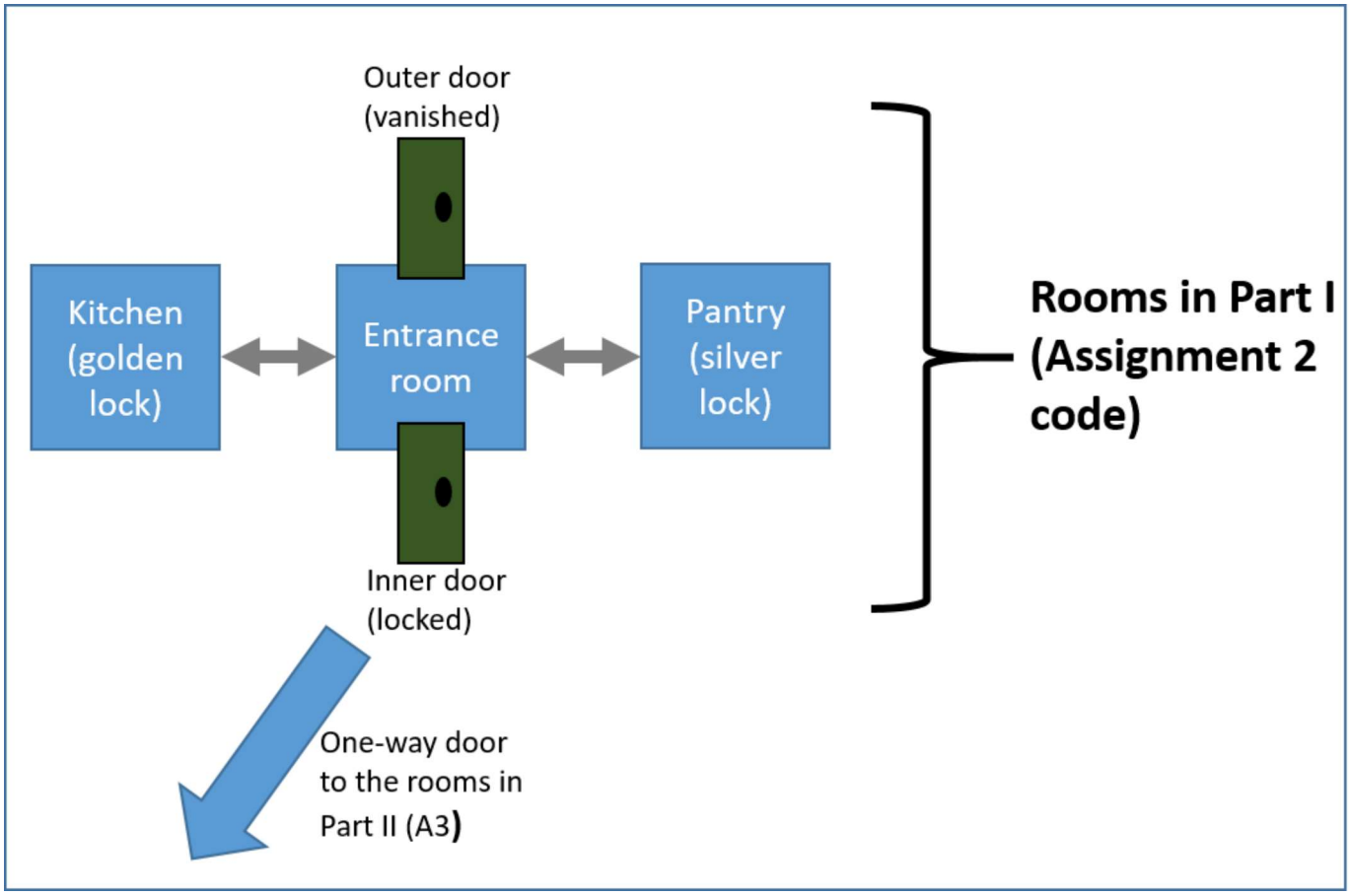
This program includes everything needed for the program to perform the actions specified in the previous assignment (e.g. display menus, get user input, verify user input etc.) but adds three additional rooms. While the player can still play the first part of this program (the A2 code can still be interacted with) **your solution must set the two locks in a state such that the inner door is unlocked (silver lock = right, gold lock = left)**. This will speed up your testing and the TA marking.

All instructions must be enclosed within the body of a function. The exceptions could include: import statements (not really needed for this assignment), the creation of global constants and the call to the initial start or main function. You will be penalized heavily if functions are not used or improperly used. A solution for the previous assignment will be posted in the [display case] on the 7th floor of ICT (near ICT 704-705) if you didn't fully complete the assignment. It's recommended that you don't take the entire solution and just include it in your program. Instead you should just use the solution to help you understand how to produce your own solution. You can find sample outputs of my solution to this assignment under:
`/home/231/assignments/assignment3/outputs`

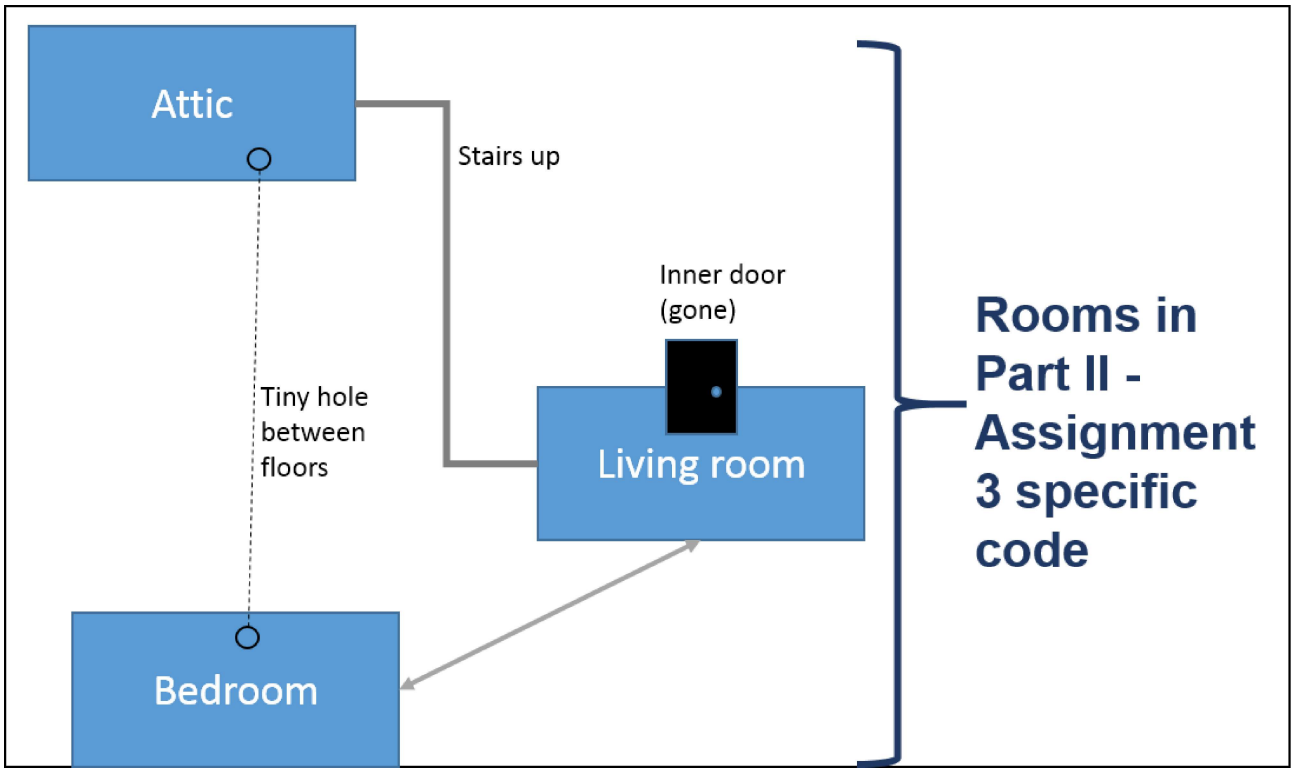
New: for those of you who want to get a head start on the assignment over the weekend: I'll post a solution and tape it to inside of the security doors on the 7th floor of ICT. (Turn left when you take the elevators in ICT).

Part I:

Your first part of this program is identical to the previous assignment except your solution will be implemented into separate functions. Once the player passes through the locked door into Part II there is no way to return the original three rooms in Part I



Part II:



Living room contents: a pot of soil, stairs going up, a dark entranceway, a ball of a string

- Viewing the pot of soil: 1) Default state when viewed, it looks dry 2) After it's been fertilized by the mouse (bedroom) when the soil is viewed a vine will grow that takes the player to paradise (game won and the program

ends).
- Stairs going up: takes the player to the attic.
- Dark entranceway: takes the player into the bedroom.
- Ball of string: the player can pick up this object. There is only one ball of string, once it has been picked up it should not show up again in this room.

Attic contents: a tiny hole in the floor, an unlimited supply of cheese, stairs going down.

- Hole: 1) player can pick up some cheese and try to drop it down the hole, the game indicates that the cheese is too big 2) If the player has the string then an option is provided to drop the string down the hole. When this occurs the string should be removed from the player's inventory (the string doesn't reappear anywhere in the game, nor does the option to drop the string down the hole appear in this room).
- Cheese: player can pick some up, the supply is unlimited so repeatedly picking up the cheese won't diminish the supply (or change the room description). However the game should track that the player is carrying cheese (this will allow the player to feed the mouse the cheese).
- Stairs going down: takes the player back down to the living room.

Bedroom contents: a tomcat which is intently watching a mouse hole (default), mouse (after cat leaves).

- Tomcat: when the player isn't carrying the string no interaction is displayed. When the player is carrying the string then there is a option to play with the cat using the string whereby the cat looks briefly at the player and then goes back to watching the mouse hole (the motion isn't enough to distract the cat).
- Mouse: after the player drops the string down the hole from the attic the large motion distracts the cat sufficiently so that it leaves the room. The mouse then comes out of it's hole. Only if the player is carrying some cheese will the game display an option to feed the cheese to the mouse. When this occurs the mouse will leave the room and fertilize the pot of soil. (The vine will grow and the player will only wine the game when the soil is viewed again from the living room). After fertilizing the soil the mouse will return to the room so there is the possibility of the player repeatedly feeding the mouse. If the player hasn't picked up any cheese then the room description should show that the mouse is in the room but there is no option to interact with it.
- Dark entranceway: takes the player into the living room.

As was the case with the previous assignment each room will provide a description of the contents, display a menu of options (which varies depending upon the actions of the player), get and error check the player's selection as long as the player remains in the room. (In the case of the bedroom the menu may stop displaying after the player has won the game).

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 and student ID)
- Header documentation: summarize functionality of program
- Documentation for each function: list the program functionality implemented in the function. Also each function should specify the type and number of arguments as well as the return type or types for the function. (In the case of a tuple describe the type of each element of the tuple e.g. (int, string, Boolean). Also any program preconditions (or assumptions about the arguments) should be documented (e.g. integer arguments are greater than zero, string arguments never consist of empty strings).
- 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)

Principles of good design for functions (most of it is a synopsis from the decomposition notes)

- Functions are one screen in length (normal screen resolution say ~30 lines max)
- Functions implement one well defined task (e.g. `processLivingRoomCommands()` vs. `processlivingRoomRunIntroductionRunConclusion()`)

- Code in one function is not duplicated in another function (not in the notes but this is just common sense that you don't write two functions where there's overlapping code - the overlap should likely be take out of both functions and moved to another separate function). In this assignment there may appear to be some overlap (e.g. each room displays a menu of options but the specific options displayed will not be identical).
- No global variables (unless you have a compelling reason that you have justified to a course instructor).
- As mentioned your program should consist of at least 7 functions including the starting function.

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.