Megan Fanning
CS 162 Rooker

Final

Theme

You are an aging showgirl, and you've had just about enough of these magicians and illusionists, you know how to stack a deck and get cut in half with the best of them, it's time they let you in on their secrets. The magician you're working for has told you to meet him at the warehouse he practices at and says he'll teach you magic, IF you can successfully identify magic and illusion.

Upon entering his warehouse you find yourself in a hall of mirrors where you must find an exit. In the next room there is a merry go round that is in the center of four rooms, use the merry go round to enter all four rooms and collect the props for tonight's show. Each of the four rooms opens to another room, some of the entrances require a key to open and some just keen observation.

The acts for tonight are sawing the lady in half, the underwater escape and a bullet catch trick with a live gun. Collect to the props needed for each trick without getting fooled by the illusions.

The show is at 8pm tonight but it's already 7:30pm, can you escape the maze of illusions in time?

	2∨ Under the stage (exit) locked	
0∨ Water tank handcuff keys* “locked” by tigers	3 ^∨ Stage hacksaw* (locked trap door)	7∨ Gun with blanks* perfume deck of cards
^> 1 Tigers + skeletonkey*	4<^>∨ Carousel	<^ 8 Bunnies (unlock tigers) scarves fake leg
∨^<> used for links number is location in vector * indicates item is required to win	< ^ >∨ 5 Mirrored room	<^ 9 mirrored room
White- classic room dark gray -locked light gray- mirrored pink - carousel	^6 Hall of Mirrors	

Design

Room Class: has a vector of the objects in the room, 4 pointers (north,south,east,west) and a type. functions: print room label, print exits, additem, remove item, announce items. The room is constructed using links and dynamic memory allocation and stored inside of a vector (to ensure proper deletion of dynamically allocated memory).

LockedRoom class: has an int which stores the address of the adjacent locked room in the vector. functions: lockChecker, determines if the user has the key to the room in their inventory.

Mirrored Room Class: has a overridden print room label method which does not list the exits to the room so the user has to guess the correct direction.

Carousel Class: every time the user enters a room the pointers on the carousel are shifted clockwise (north now points east). This is implemented using a counter in main, when the carousel room is accessed a turn function must be called to access which direction is linked where.
Carousel overrides the superclasses get room function so instead of returning the north pointer it returns a pointer to the east (if the room has rotated once).

Inventory Class: contains a vector of items. Each Item has an id number which can unlock doors or the win condition, as well as a string to tell the user what the item is.
functions: printBag, bagFull (returns a boolean), emptyBag removes a item of the user's choice from the bag, itemManager which manages the user's input and allows them to access each of the previous functions.

Testing

I went through every room in the maze as my first test, which allowed me to find design flaws (like doorways not being locked) and confirm that all the links were working. Once that was completed, I turned on the carousel code to test if that was working. I found that my mirrored rooms code and my carousel code wasn't working as expected because I failed to make the super class functions virtual initially, and the super class implementations weren't being overridden.

I attempted to pick up more items than allowed by the bag's size limit. I checked object permanency (picking up items removed them from the room, and when an item was dropped it was added to the room). I tested the win and lose conditions of the maze.
In debugging I found that I was having a lot of trouble trying to assign the link for the locked rooms only to realize I was passing in a vector of Room not Room*.

Reflections

The Carousel was one of the most challenging parts to design. I found on this design I spent almost double the usual amount of time I spend on design, which made doing the actual code much faster. However, when I did run into issues it become more difficult to be flexible/alter my design to add functionality that I found I needed but hadn't planned for (for example, I didn't plan out how the user would use the items because initially I planned to just have the room determine if the key was present - but I realized that would be confusing for the user.)

Once I completed my rooms I realized I had to downcast my currentRoom pointer to the various subclass types (LockedRoom, Carousel), in order to gain access to the subclass unique methods, which was frustrating. I realized I should have had a virtual doRoomFeature function in the base class which I could have overridden in the subclasses. This would have also avoided having to add a room type field.

I ended up ditching my idea of having something make the labels visible for the mirrored

rooms, since there were only three mirrored rooms. If I expanded my maze I might have felt like it was more necessary, but it seemed like it would actually take away from the player's experience rather than add to it (also it would make the design of the mirrored rooms and the locked rooms almost the same). So I opted to remove the perfume bottle/sprayer revealing the exits

I should have used a bool for linking the locked rooms once the key was accessed. Several of my design flaws became apparent in testing when I found bad code "smells" like having to provide a type, having to cast my current room pointer to the subclass (to access Carousel and LockedRoom's functions) and the need to repeat the unlocking interaction every time a player accessed a room.

I had a lot of fun with this project and think I would like to redesign it again to refactor and make it better, now that I've learned some from my initial mistakes.