Brianna Carter
CS 162 Final Project
"Betrayal on the Alquacir"
6/9/15

My first step in design was to create a concept that would be fun and realistic for me to create and meet the requirements. I decided to use the story of a Dungeons and Dragons character I created for inspiration. "Betrayal on the Alquacir" is about a teenaged Elf named Glingaerwen whose father is a well-known captain. It takes place on his ship Alquacir where tragedy struck during a storm and your father fell overboard. You know that it wasn't an accident so you navigate the ship's rooms searching for clues and talking with some crewmembers.

I started with designing the simplest elements and building up. I created the player and item classes based on a shopping cart assignment I did in 161. The player needed to keep track of the clues the user finds so I needed a vector of item objects. The items needed to have a name and a description so the user would know their benefit to the story. There needed to be functions for adding, removing, and getting information. I tested these individually using simple code adding, removing, and displaying all of the items I would need in various orders.

Next I worked on the ship and room class. I made a list of all of the rooms as well as what would be in the pointers for each room. I knew the player would start in the hallway.

Hallway
N: Deck
S: Holding cell
E: Supply Storage
W: Captain's quarters

Captain's Quarters
N: Painting (interaction)
S: Captain's Log(item)
E: Hallway
W: Window(Interaction)

Deck
N: Bird's Nest
S: Hallway
E: Navigation Wheel
W: Stern

Bird's Nest
N: Sky(Interaction)
S: Deck
E: Broken Arrow(Item)
W: Sky(Interaction)

Navigation Wheel
N: Bow
S: Deck
E: First Mate(Interaction)
W: Black Ink Residue(Item)

Holding Cell
N: Hallway
S: Mess Hall
E: Guard (Interaction)
W: Wall(Interaction)

Storage
N: Weapons(Interaction)
S: Mess Hall
E: Gun Powder(Interaction)
W: Hallway

Mess Hall
N: Storage
S: Cook(Interaction)
E: Crew quarters (interaction)
W: Poker(Interaction)

I needed the rooms to contain a structure of pointers so the user could navigate from this room. The functions would need to add the links, display those

links, display the current room information, and choose the direction to move. The ship class needed to access the rooms, I thought of them similar to the player and item classes. I needed to keep track of the current room.

The best way to approach this part of the design was to ignore the navigation requirement and focus on the ship creating the room and the linked list being created correctly.

I used chapter 17 and lab 8 to design the linked list in the room class. Values 1-4 represent north, south, east and west, respectively, so that value could be compared to a user choice from a menu and know which direction to move. The linked list pointer is set to the first node to access all of the nodes.

At first I used strings instead of pointers in the linked list so I could test without all of the rooms. The add function was overridden in the child class and four rooms were passed as parameters and new nodes are created. I tested this by creating a hallway object, adding the linked rooms, then accessing and displaying each link's information.

The most difficult part was creating a way for the player to move through the rooms. I ignored the items and just looked at how to keep track of the current room. I decided to use a room pointer in my ship class to keep track of where the player is. I changed the strings in my structure to room pointers so could assign the current pointer to the link pointer the user choose. I added a function called get current name that returns the pointer to the room the player wants to move into. This function calls a room function for the current room that will display the room information. I added a function in room called choose direction that displays the linked rooms, allows the user to choose the direction. I ran into trouble with core dumping because some of the room pointers are null. Those represent interactions and objects, so I needed to use a catch for when the user chooses one of the null pointers. I decided to use if and else if statements to indicate that the current room is changing or to display the interaction at that location This was easy to accomplish with the value representing the direction. All I had to do was compare the value in each node to the user's choice. When a pointer is null, there is nothing to display so the usual method of getting the information held in the pointer wasn't working. In those instances I had the interaction text in the if statement.

Adding the items was last to be implemented. After thinking, the simplest way would be to use Boolean variables in the room classes where the items are located. The variable would be initialized to false in the add function. If the user chooses the item interaction the variable is set to true. I added three get functions to my ship class that returns the Boolean value of the item. If the value is true, the player add function for that item is called. Since the Boolean values are checked every time the room is changed, if a user enters a room more than once and interacts with that item, I didn't want the item to be added more than once. To solve this I added a loop in the player function to search the vector for the item and only adds it if it isn't present. When the item is added, the item count increases. The main function is in a do/while loop until the item count is three. The program then calls the solve function to allow the user to choose their answer.

Testing was an ongoing process. It was made easier by splitting up the design process into small problems that I had solved in previous assignments. One problem

I faced was a core dump when I was in a room that included interactions. I was able to get the hallway working as it should but when I moved to the captain's quarters, it wouldn't display the links. Since I gradually added elements I was able to easily look at the differences between the two classes and figure out that the captain's quarters linked list had null pointers that couldn't be printed. It was a simple fix to display the interaction message. I also ran into an infinite loop when I was trying to compare the value in the linked list to the value of the user's room choice. I added some cout statements and discovered that I stated while(ptr) instead of while(ptr->next != null).

My testing was very straightforward since I modeled a lot of the code after code I had already written. Designing and writing code in small segments is a much more efficient process than trying to tackle the whole project at once. Usually I try to figure out the big picture before looking at the elements and would get confused trying to juggle everything. Being able to work on small tasks made this program come together quickly. I also wrote out the relationships between the classes and looked at composition and how things needed to interact. At each stage I made a list of tasks to be accomplished and focused on one task at a time. This method allowed me to stay calm and stress-free so I could remember what I learned quickly. Making lists and charts is helpful to visualize what should happen and makes troubleshooting faster and easier.

Testing

	Expected Output	Result
Player/Item class	"There is nothing in your sack" "Added broken arrow to your sack" "Added black residue to your sack" "Added Captain's log to your sack" "Your sack contains: Broken Arrow Black residue Captain's log" "Removing Broken arrow" "Removing Captain's log" "Your sack contains: Black residue"	Got the expected output the first try since I was using code I had previously written.
Ship/Room class		
Testing if the links were assigned properly.	Room Name: Hallway North: Deck South: Holding Cell East: Storage West: Captain's Quarters	I also used code I had written previously for this function so the links were assigned easily.

Changing rooms from Hallway	You can move to these rooms: 1. North: Deck 2. South: Holding Cell 3. East: Storage 4. West: Captain's Quarters Please make a choice: 4 Moving to Captain's Quarters Captain's Quarters: (description) Here are your choices: 1. North: There is a painting on the wall 2. South: The captain's log is on the desk. 3. East: Hallway 4. Window	The program stalled after the user made their choice. I put some cout statements to find what was happening and realized I created an infinite loop and needed to change my parameter in my while loop. This corrected that problem. Next I got a core dump after the current room was set to captain's quarters. It wouldn't display the linked room information. I realized it was because this room has interactions and not just rooms so I changed the output for the interactions from the pointer information to a phrase describing the interaction.
Collecting the Captain's log	Added Captain's Log to sack	This is displayed only after the user moves out of the quarters. It doesn't display right after the interaction. This is how it was designed.
Displaying the solution when 3 items are collected	"You have collected all of the clues! Choose the killer: 1. 2. 3. 4. 5.	The user has to move out of the room where the final object is found before the solution is displayed. This is similar to collecting items.