Sanjo Abraham
Final Project Reflection

Test Table:
Assume proper int validation.

Move Places	Find Item	Buy Item	Sell Item	Repair	Actions	Outcome
Move up, down, left, right						Move properly, change board to new space type if it is type “fog”. Won’t allow player to access out of array
	Find key/regular item					Move item into it’s containter (vector). Can access this info every turn
		Buy usable item				Move item into it’s container(vector). Can see this info every turn. Requires you have adequate gold
			Sell non-key items			Deletes the item, and removes it from the container. Increases gold bar count.
				Repair		Removes the item from it’s vector container, but sets a bool as true, so game know’s that to not find it again.Requires that you have adequate gold for repair work

					@Nebula	Decreases shields until you repair matter stabilizer
					@Debris	Can search, do nothing, or use item. Occasional damage taken from space junk.
					@Station	Can repair, trade, or take a sabbatical(increase morale).
					@Planet	Can repair, trade, take a sabbatical(increase morale), or talk (usefulness of talk is not implemented yet)

Design:
I spent a lot of time on the design portion of this project. I brainstormed several idea, some of which, you can see incomplete pieces of within the code, for example, the enemy spawn class is not used, but I intended to add that as another challenge. The reason that many of these functions were not implemented was due to time constraints.
As far as how I designed the program, I created a dynamic 2D array of space pointers, which could use polymorphism to change into various derived classes. The player would choose the size of the array, and their name, as well as if they would like to have faster gameplay or not (in terms of text speed). I implemented a slow-read class, which would read individual characters, as a style choice. Next, I implemented various functions, some of which could only be done on certain spaces. I opted to have the Game.cpp file be the main source of functions. It would read the type of space, and list the possible actions that one could preform on that space. Next, items were added using polymorphism as well. The items would be stored in vectors as the player gained access to them, and players could view what items that they currently have at the beginning of each round. There were several design choices I made, such as modifying some functions to be boxed off using a divider function. Another thing that was done was allowing players to keep trading/repairing in a loop until they exit. I did so because I felt that it made most sense to not only allow them to have one sale/trade per move and prevent them from frustratingly going back and forth between menus.

Reflection:
Overall, I felt that this assignment was well worth the effort, and a very good example of what I have learned in this class. Many of the problems that I encountered during this project were

easily remedied, my most time consuming task was creating a game that made sense, hence the large amount of time spent during design. Drawing diagrams was the most helpful tool I had during that process. In the beginning of the project I had a difficult time deciding out to implement the item system, since I had so many items, which all had different functionality. I opted to create multiple files for each item, and have them work in a polymorphic fashion, but I still wonder if that was the best option. I believe that I could have created a struct within the Game class, and created a vector that contains said struct, and have a function that adds all the items into those containers. That certainly would have helped with the bulkiness of the project. Another approach that I attempted was to use subdirectories in order to divide the items from the rest of the files. I tried using recursive makefiles and the “wildcard” function, but it would often lead me to errors.