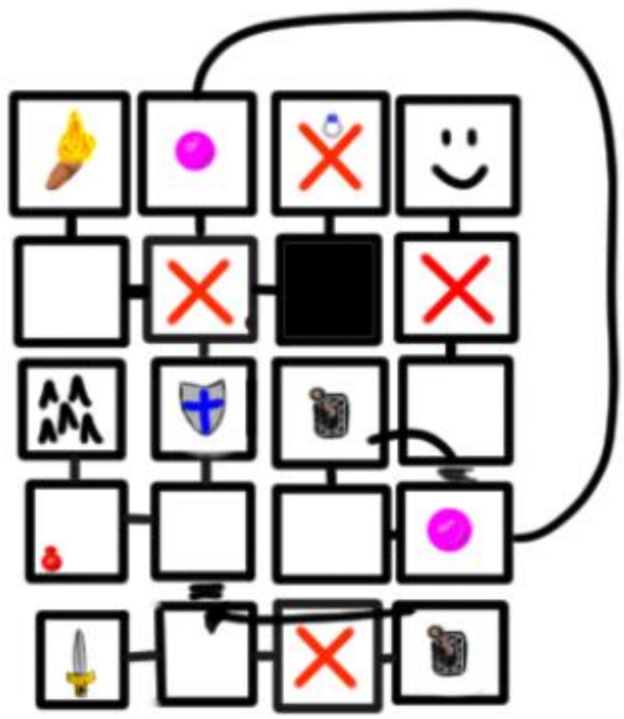
Understanding:

The goal of this project is to demonstrate a comprehensive overview of the information we have learned throughout the semester. We will be designing a maze structure with at least ten different rooms. The player will need to be able to navigate through the rooms of the maze, and we need a method of keeping track of the player's location throughout the maze. Each room needs to have 4 pointers that either point to an adjacent room or NULL of there is a wall present. The room class itself will be pure virtual and there need to be a number of children classes derived from it that create the actual room objects in the program. The player must have a goal, or winning condition, known to them to achieve victory in the game, as well as a timer to ensure they are prompted along their journey. The player needs to be able to interact with the room in some ways such as moving a lever to make a false wall open or be able to pick up items from the room. There is a bag requirement in that the player must be able to hold a certain number of items before their bag become full. Movement should be done in the North, South, East, West directions.

Design:

Upon reading the requirements the very first thing I did was physically drew a map of how I wanted the game to work out, shown below. The player will start at the bottom row between the sword and the X. Any X represents a monster. The sword represents the sword item that the player will acquire to be able to progress past the starting monster. My plan it to allow the player a slim chance to defeat any monster if they have not acquired the necessary item and otherwise a very high (90%+) chance. After acquiring the sword, the player will progress past the monster to the switch on the bottom right. This will open up a false wall in the room they started in allowing them to continue. They will then be able to move up and to the left or straight up. If they move up and to the left, they will first be able to find a potion which increases their maximum health, and then later run in to another switch. If they hit this switch however, they will fall to their death and be restarted at the beginning of the dungeon. If they continue moving North, they will pass an armor room and if they search it they will be able to find the shield. If they continue North without finding the shield they will likely die to the monster in the next room, but if they have the shield the majority of the time they will be ok. Upon defeating the monster, they will be able to either move East to the dark room, North to the orb or West towards the torch room. If they attempt to walk through the dark room without first acquiring the torch, they will fall to their death. If they do have the torch, they will then be able to pass through the room and fight the monster in the next room. This monster will drop a ring which protects the user. They will then move to the teleport orb, move up and to the left to the switch, hit the switch which opens a false door in the teleport room, and move north form the teleport to fight the final monster and win the dungeon.

I plan to design a room class (which will then be brought over from the previous assignment) as well as the monster class (which will also be brought over from a previous assignment). The room class will be made pure virtual and I will have these subclasses for the rooms: normal, monster, item, dark, trap, and switch. The monster type room will need to contain a monster type pointer. Item rooms will need to contain items for the user to acquire, this can either be done with a class/structure, or possibly just by using a Boolean to set their item status equal to. The dark type room will need to have a condition where if the user tries to move any direction except for West (if they do not have the torch) then they will fall and die, but if they do have the torch they will be able to successfully navigate it. I am currently considering two options: The first being a physical clock that informs the user of the starting time, and each time they enter a room they will be given a countdown until the game ends. However, the option I am more leaning to is making the countdown based on the number of moves the user makes as this is more in line with previous MUDS I have played and I think will make for a better playing experience. The switch rooms and teleport rooms are going to need to contain an addition 5th pointer to a room that allows them to interact with a far away room altering its characteristics.



My second point of design after my intial map and plan was to determine a good stat build for the player versus the monsters. I decided to use my combat simulator with 10,000 battles between a barbarian from the previous assignment (with the player's stats) and a blueman from the previous assignment (with the monster's stats). Shown below is the progression chart I have designed by trying a number of setups between the two parties. Shown on next page.

	Skeleton: Stats 3 Armor , 20 HP, 2 attack die, 1 defend die	Living Armor: Stats 3 Armor, 20 HP, 4 attack die, 2 defend die	Blood Golem: Stats 3 Armor, 20 HP, 4 attack die, 2 defend die	Vampire: Stats 3 Armor, 20 HP, 8 attack die, 2 defend die
Player (no sword): Stats 1 Armor, 20 HP, 2 attack die, 1 defend die	15% win ratio			
Player with sword: Add 2 attack dice roll (total of 4 attack dice)	93% win ratio	14%	14%	
Player with sword and armor: Add 2 armor dice roll (total fo 4 armor dice roll)		91% win ratio	91% win ratio	9%
Player with sword, armor and ring (hamstrings the enemy, acts as a protection trinket)				91%

This allows the player to progress, and if they acquire the necessary items they will nearly always win the combat, but there is a slim chance they may win without the item (or a slim chance they may lose with them). I think ratios of 100% win/loss would make for less exciting “edge of your seat” combat so I was glad to come upon stats that felt like this.

Testing:

Testing was done on this project in an incremental fashion. I first decided to create a dungeon completely of normal rooms, which I then tested thoroughly to ensure they worked before moving on to adding an additional type room. I then added the monster room next, as I considered this complicated and wanted to try to debug any monster room issues before the program was more complex. From here I then added in item rooms to the existing map, which caused problems with their interaction with the monster room. I then added in switch rooms to the working version, and finally added in the teleport room. Major test issues documented below (minor ones too numerous to reasonably show here).

Input	Output	Expected Output
attempt to enter monster in to monster room	text stats that the room is empty	a monster should appear and attack the player
add in item room to working dungeon with only normal and monster rooms	monster previously present is now gone	the player should be able to acquire items and fight monsters
player fights with the monster	the same numbers appear every single time	combat should be randomized each time
move into room of previously defeated monster	Monster combat begins again	should run another set of statements once monster is defeated
move in to room where monster previously defeated you	defeat dialog for monster runs	should treat the room as if it were new
walking in to room with a switch	automatic core dump	should allow the user the switch option or to leave the room
add item rooms in to the main game	all of a sudden all rooms display the dialog about being in the first item room	should display a message based on the room the player is in
player moves to left and finds first item	once they return right to starting room, they are stuck there	should be able to continue moving through the maze
player defeats skeleton in tile 3, 4	moving to right from this tile does not move their location	it should shift them to the bottom right tile
falling in dark room	gives message of room above the starting room after death is resolved	should display text from the starting room

fights final monster with ring	player is 1 shot	ring should hamstring the enemy

Reflection:

I can't believe I just made this. That's my original reflection. Another interesting point I've noticed: working on this DOES NOT feel like work, I really truly enjoy doing this, and in retrospect I'm looking back on the past week and realizing how much time I've spent doing this is crazy. Anyhow, back to the design:

I didn't actually need as many things as I originally thought I did, which was great. I ended up not needed an additional pointer in the switch rooms, but instead just gave them an integer variable which represented the room they were going to alter. A series of if statements was then run and based on the value of the integer they changed the rooms to be altered by the switch. I thought I would need a different type of room for the dark room, but in the movement area of my code I just put an if statement that if they attempted to move any direction but west in the specific room without a torch it reset them to the beginning. This ended up causing another issue in that the next statements below it were the movement statements, and the player was immediately moved to the start AND THEN moved from that location, so I needed to add in a bool to let the program know the player fell and to not execute subsequent movement statements if they did.

Regarding the timer, I finally decided that only one condition would move the time forward: the player actually making a choice to move. I wasn't sure what a "reasonable" time frame was, so I did two separate tests here: I run through the dungeon myself and counted the number of times it would take it the player made every mistake once, and let my fiancé play through the game knowing nothing about how to beat it. I eventually game to the conclusion that allowing 48 moves seemed like a fair number that makes the player feel pressured without making the time frame seem unfairly short.

Regarding the items and bags: I originally thought it would be cool to add a potion in to the game to increase the player's maximum life. Interestingly, changing their life had very little effect on the simulations I ran (compared to changing any other variable) so I decided that 4 items was enough and scrapped the idea. I originally thought "I should make some sort of physical item in the code such as a structure to represent the bag" but I eventually decided it was unnecessary due to the way I designed the game. There were 4 items in the game and I placed text informing the user anytime they acquired all of them that their bags were full, which I believe met the requirement.