Requirements
Create an old style adventure text game.
It must have

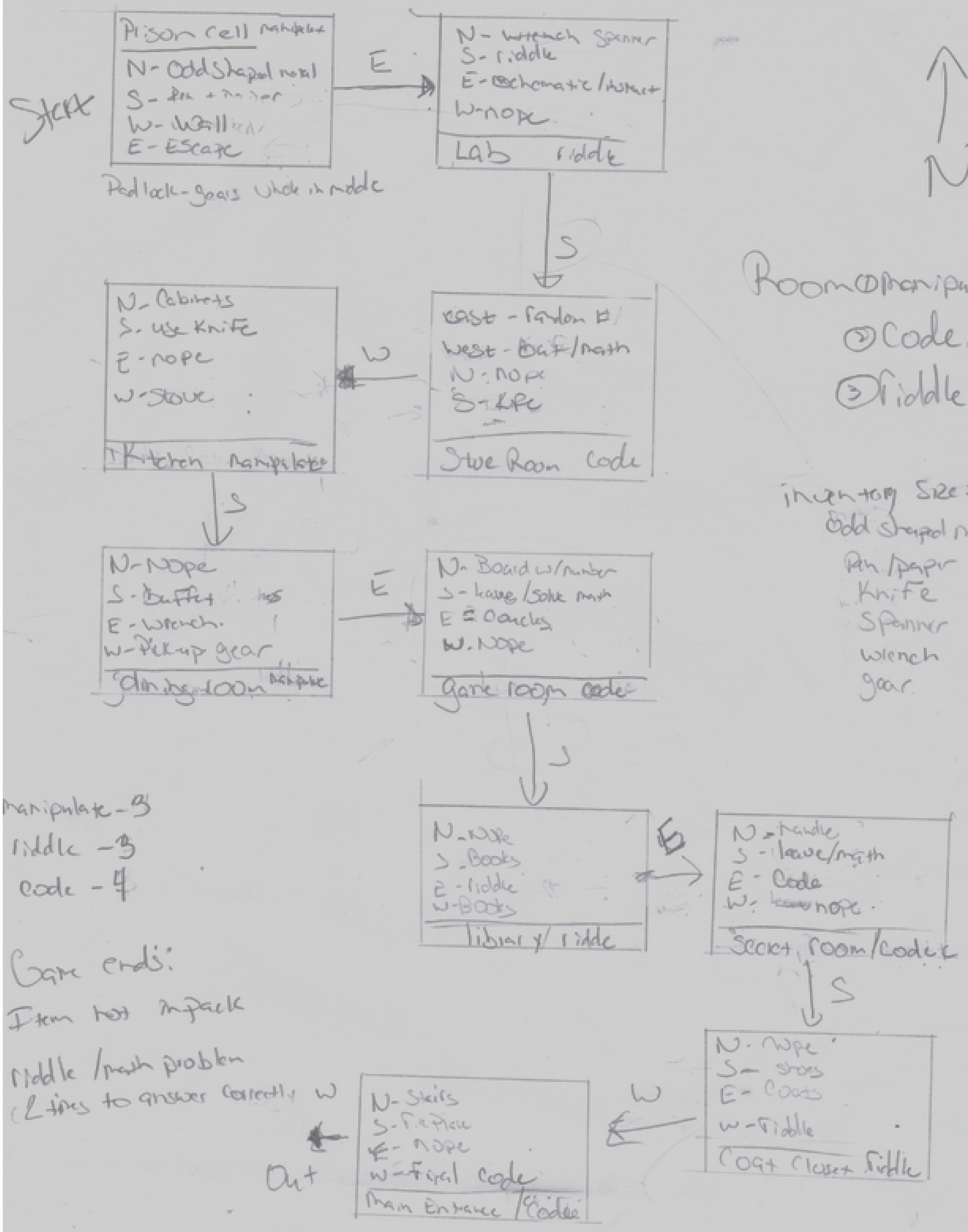
- struct for room pointers
- limited inventory space
- timer to get player to move along
- player must do something in the rooms
- must have 10 rooms
- the rooms will have a parent class that is not instantiated that they will be derived from

Design:
<u>For basic movement:</u>
Prompt player for n/s/e/w direction.
Call appropriate movement function.
Room location on map determines what actions will occur.

<u>For riddles:</u>
Print riddle for user.
Inform user they have 2 tries at the answer.
If try is 2 or 1 and correct, moveOn bool is set to true.
Else player is captured and the game ends.

<u>For manipulating doors:</u>
Search users inventory for item from previous room.
User has option to bypass all objects but the main cell room key.
If object is in inventory, user will take it out and perform action and return it to pouch.
Else if object is not in inventory, player is captured and game ends.

<u>For final room:</u>
User must collect 3 numbers that are scattered throughout the 10 rooms. The numbers are randomly generated for each room and game.
As with riddles the user gets 2 chances to input the correct answer and escape.

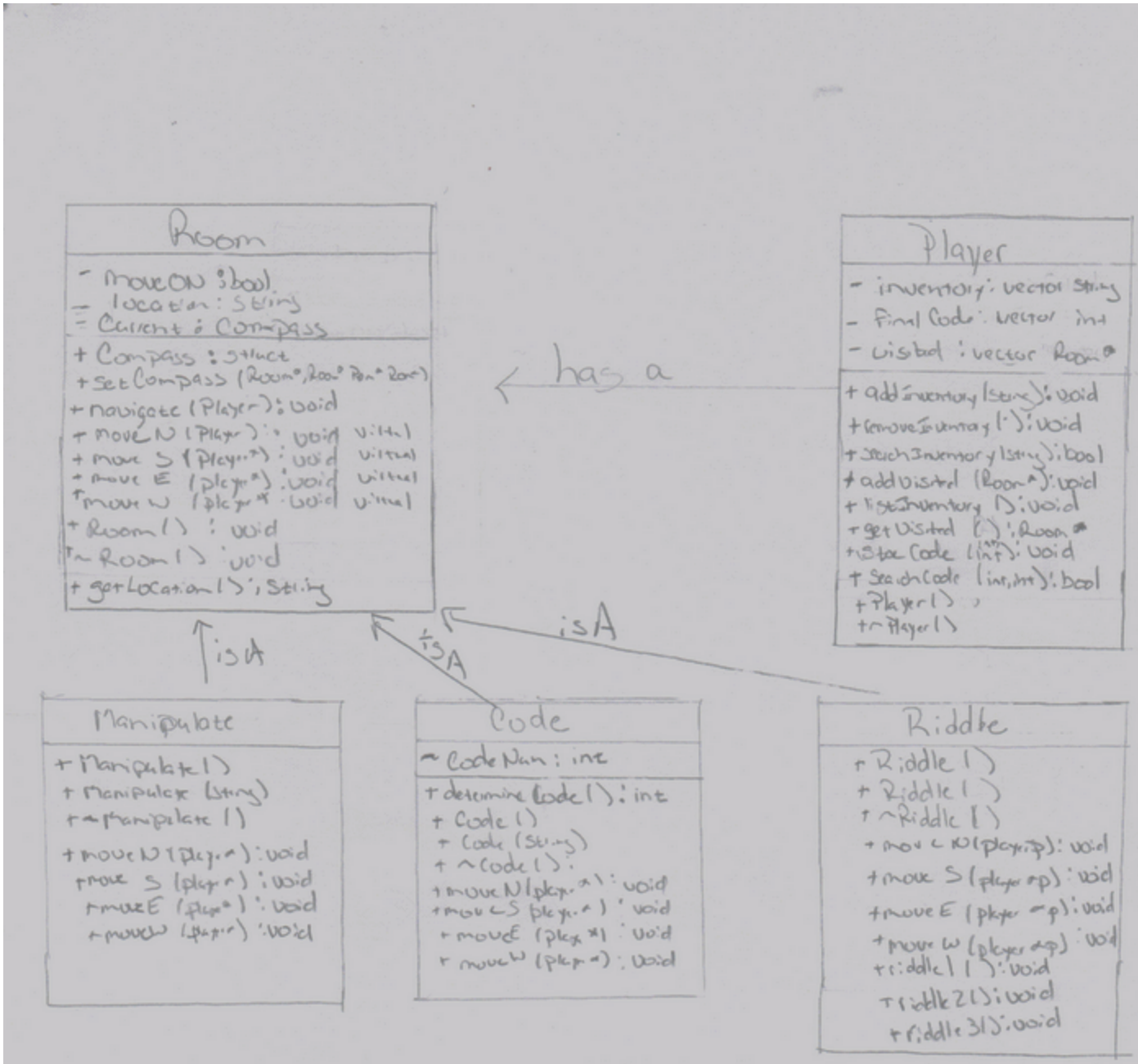


Room ① manipulate
② Code / math
③ riddle

inventory Size = 4
Odd Shaped metal
Pen / paper
Knife
Spanner
wrench
gear

manipulate - 3
riddle - 3
code - 4

Game ends:
Item not in pack
riddle / math problem
(2 tries to answer correctly) w


Room
- moveON :bool
- location : String
- Current : Compass
+ Compass : struct
+ setCompass (Room*, Room* Room* Room*)
+ navigate (Player) : void
+ moveN (Player) : void virtual
+ moveS (Player*) : void virtual
+ moveE (player*) : void virtual
+ moveW (player*) : void virtual
+ Room ( ) : void
+ ~Room ( ) : void
+ getLocation ( ) : String
has a
Player
- inventory : vector String
- FinalCode : vector int
- visited : vector Room*
+ addInventory (String) : void
+ removeInventory ( ) : void
+ searchInventory (String) : bool
+ addVisited (Room*) : void
+ listInventory ( ) : void
+ getVisited ( ) : Room*
+ storeCode (int) : void
+ searchCode (int, int) : bool
+ Player ( )
+ ~Player ( )
isA
isA
isA
Manipulate
+ Manipulate ( )
+ Manipulate (String)
+ ~Manipulate ( )
+ moveN (player*) : void
+ moveS (player*) : void
+ moveE (player*) : void
+ moveW (player*) : void
Code
- CodeNum : int
+ determineCode ( ) : int
+ Code ( )
+ Code (String)
+ ~Code ( )
+ moveN (player*) : void
+ moveS (player*) : void
+ moveE (player*) : void
+ moveW (player*) : void
Riddle
+ Riddle ( )
+ Riddle ( )
+ ~Riddle ( )
+ moveN (player*) : void
+ moveS (player*) : void
+ moveE (player*) : void
+ moveW (player*) : void
+ riddle1 ( ) : void
+ riddle2 ( ) : void
+ riddle3 ( ) : void

Testing Plan

Test 1: walkthrough with correct directions and actions
Input:
cell: n- metal s-pen/paper e- leave room manipulate metal
lab: n - spanner s- solve riddle (stars)
store: s - knife e - code (random) - w riddle(3)
kitchen: s - use knife
dining: e - use spanner
game: n - code (random), s - solve riddle (2)
library: e - solve riddle (shoe)
secret: e - code (random), s - solve riddle (4)
coat: w - riddle (candle)
entrance: w - door (enter the 3 random number from room store,game,secret)
driver function: main()
expected result: win the game
actual result: with the game

Test 2: walkthrough pick up more item than inventory
Input:
At dining room pick up gear.
driver function: main()
Expected results: prompted to drop item or bypass it. if drop, an item is removed and new added.
Actual results: prompted to drop item or bypass it. Dropped item and added new.

Test 3: walkthrough answer riddle wrong
Input: Answer lab riddle wrong 2 times
driver function: main()
Expected outcome: player captured. game ends.
Actual outcome: player captured. game ends.

Test 4: walkthrough answer final code wrong
Input: Input final code wong 2 times.
driver function:main ()
Expected outcome: player captured. game ends.
Actual outcome: player captured. game ends.

Problems

The first problem encountered this week was that sometimes help is no help or is more confusing than what you originally thought. Some of the final project requirements were a little confusing nature, however, when going to the discussion board to see about clearing the confusion ended up even more confused. In the end had to ignore the discussion boards and interpret the requirements how I understood them. To be honest, I am not confident that I have

interpreted them correctly but after re-writing just about everything I had done three times I feel I have done a decent job. It is a learning experience in that in a job a customer may give you vague and misleading requirements and asking colleagues can compound the problem. It points out that it is important to ask the right questions to clear up any confusion.

The second problem encountered was including the Player.hpp in with the Room.hpp. The compiler gave me odd messages about invalid template. I remembered that you could forward a class in another classes header file but was under the impression that include the hpp or using the forward declaration were interchangeable. This was a huge misconception. In this case since the player class was being passed as a pointer, forward declaring the class was a better method and it removed the compiler error message. A problem like this points out there is so much more to C++ then what we learn in the first 20 weeks and you need the initiative to go and learn on your own. The flip side is that finding useful and understandable information is like finding a needle in a haystack.

Out of all the programs I have tackled in the last 20 weeks this is the one I am the least proud to produce. I need to be less stubborn about asking for help from both the TA's and the instructor.