Patrick Mullaney
ONID: mullanep
TA: Soo-Min Yoo

Final Project.

Understanding:
Design a text base game using OOP, inheritance, polymorphism, and pointers that satisfies requirements.

Requirements and implementation:
-Goal for your player:
    -Survive traversing the map until you reach the Kingdom and defeat the final boss, Negan.

-Abstract space class with at least 6 spaces of 3 different types (derived classes). Each space will have a special action:
    -Parent class Space represents spaces to fill the map/grid.
    -5 total: The Kingdom, Hilltop, Armory, Horde, Prison, and negansmen.

-4 pointer variables that link to other spaces:
    -Each space has four space pointers: top, bottom, left, and right, which point to adjacent spaces on the grid. The grid is not a wraparound by design choice and has boundaries.

-Way to keep track of your player location:
    -A space pointer location will keep track of where the player is.
-Container to keep track of items (with specific size):
    -Satchel is an Item array of size 2 (specifically set at size 2 so as to require player to make a choice between items that will affect the outcome of the game).

-1 or more items will be required as part of the solution:
    -Each of the three items that can go in the satchel, (food, guns, key), contribute one of the three solutions respectively.

-Time limit:
    -This is implemented as both a decreasing food supply (you must arrive at the kingdom before food runs out) and the fact that the player can die.

-Each player must interact with parts of the space structure.
    -At each player's entrance to the space, the virtual turn function is called, which alters the data in the space. It may involve an exchange of items, or it may involve battling another fighter. Additionally, from the beginning, "good" spaces (Kingdom, Hilltop, and the Armory), are marked on the map with specific characters. "Bad" spaces (Horde and Negansmen) are initially marked with characters to represent geographical locations on the map ('.' For a road, '#' for

grass, etc). After these spaces are “discovered” (entered by the player), when the player leaves, they are now marked on the map (‘W’ to represent a Horde of walkers and ‘S’ to represent Saviors/negansmen). Additionally, each “bad” space contains a fighter (Walker or Savior), if a Savior dies during combat, they become a walker.

-Menu was created with character selection.

<u>Design:</u>

For my theme and design, I wanted to create a program that tied together all aspects of each assignment we did throughout the term, but that I could also continue to build on as practice in the future. I chose to design a 2D dungeon crawler based on The Walking Dead. The theme is essentially Negan (boss) has surrounded The Kingdom, trapping residents, and food is running out. The player will traverse the map collecting items and battling enemies until they reach the kingdom. Once they reach the Kingdom (or if, player can die along the way), a boss battle with Negan ensues.

<u>Ways to win:</u>
As Negan is significantly stronger on his own, the player alone will likely not be enough to win (unless they have an insanely disproportionate series of lucky attacks). As such, there are three ways to increase your chance of winning:

1. Defeat a lot of Negan’s men (Saviors) along the way and take their guns. If you have enough guns you have in your possession when you arrive at the Kingdom, you can select other characters to join you for the boss battle.
2. Collect food along the way and when you arrive at the Kingdom, give all your food to King Ezekiel as a gift. He will then join the player in the final battle and has significant defense abilities.
3. Go to the armory and trade all your guns for a key. User is not told what this key is for, however when you arrive at the Kingdom, you are offered the option to give the key to King Ezekiel. If the player does, they are rewarded with a special fighter, “the Tank”, that will take the place of selected character in the final boss battle.

<u>Classes:</u>
Space (base).
The Kingdom, Hilltop, Armory, Horde, Prison, and negansmen. (derived).
Fighter(base).
Rick, Michonne, Daryl, Savior, Walker, King Ezekiel, Tank, and Negan(derived).
Grid.
Grid represents a 10x10 2D array of space pointers. Spaces are allocated dynamically as I wanted to leave open the option to change the space types based on the flow of the game. Grid also initializes the space pointers to point to their respective adjacent spaces. The grid is specifically designed to not be a wraparound.

This is a design choice as while I feel a wraparound is easier to code, I wanted to prevent the player from “cheating” and getting to The Kingdom too quickly by immediately moving toward the bottom and coming out on the top of the board near The Kingdom.

<u>Gameflow</u>

-Player will select a character, which will then be placed on the board.
-Player will then traverse the board, moving through different spaces.
-Player will interact with spaces during virtual turn() function for each space. In each turn, depending on the space, players can trade food, trade guns, or battle for items. In certain spaces, they can trade or give the key to obtain a tank.
    -Each battle within a space will be a round of attacks and defense by each player, but will not go until the death.
-Traversing the map will occur until either the player dies or reaches the Kingdom. Upon reaching the Kingdom, the player has a variety of options. The player may heal. The player may “gift” all their food to King Ezekiel, which will lead King Ezekiel to join the player in the boss fight against Negan. Or, if the player has obtained the key from the armory, they can give it to King Ezekiel to obtain the tank.
-If player has the tank, only the tank will battle Negan. Otherwise, player can select additional characters to join them in the fight. The number of additional characters they are allowed to select for the final battle is dependent on how many guns they obtained.
-The boss battle with Negan will then commence until Negan is defeated or player and all characters die (if player character dies in the boss battle, but their selected allies survive and defeat Negan, the player wins).
-Display winner at the end or if player has lost prior to this, display messages indicating game over (player died, kingdom ran out of food, etc).

<u>Fighter classes (parent class).</u>

a. Data elements:
-string type.
-int strength, armor, row, col, guns, food.
-bool key, tank.
-Item** satchel
-Space* location
-Space*** grid

b. Methods:
-getType();
-getArmor();
-getChar();
-getStrength(), setStrength();
-virtual attack()
-virtual defend();

-virtual heal();
-getKey(), setKey();
-getGuns(), setGuns();
-getLoc(), setLoc();
-getSatchel();
-addItem();
-free();

Rick class.

a. Data elements: Fighter elements, int life
b. Methods: Fighter methods.

The only deviation from the parent class is that Rick has an extra life and will be revived one time if strength is less than or equal to 0.

Michonne class.

a. Data elements: Fighter elements, no additional ones.
b. Methods: Fighter methods, no additional ones.

Michonne has a higher attack than other characters and a special ability, if she rolls an 18, her sword attack is implemented, inflicting 100 damage. She is also able to sneak by walkers, however this is implemented in the Horde class.

Daryl class:

a. Data elements: Fighter elements, no additional ones.
b. Methods: Fighter methods, no additional ones.

Like Michonne, Daryl has a special attack, Bow, that inflicts 50 damage if an attack of 18 is rolled. Additionally, Daryl heals far more than any other, 70%.

Ezekiel class:

a. Data elements: Fighter elements, no additional ones.
b. Methods: Fighter methods, no additional ones.

Ezekiel is not a selectable character, however is a player ally that will be accessible if player gives all their food to King Ezekiel as a gift when they arrive to the Kingdom. Ezekiel has a significant defense capability.

Tank class:

a. Data elemetns: Fighter elements, no additional ones.
b. Methods: Fighter methods, no additional ones.

Tank is only obtained by trading all guns for the key at the armory, then giving the key to King Ezekiel at The Kingdom. Tank class has strong attack range and high strength points.

Savior class:

a. Data elements: Fighter elements, no additional ones.
b. Methods: Fighter methods, no additional ones.

Saviors are Negan's henchmen and are not selectable players. Their methods are limited to basic attack, defend, and heal. If killed, they become walkers, however this feature is implemented in the space class.

Walker class:

a. Data elements: Fighter elements, no additional ones.
b. Methods: Fighter methods, no additional ones.

Walkers are basic fighters that occupy the Horde spaces, Prison space, and negansmen spaces(if a Savior dies). As they're already dead, they can essentially be damaged to the point that their strength is negative.

Negan class:

a. Data elements: Fighter elements, no additional ones.
b. Methods: Fighter methods, no additional ones.

Negan has all the basic elements of the fighter class, however he is initialized with significantly strong strength and has a strong defense.

<u>Space classes:</u>

a. Data elements:
   -int row, col
   -Fighter* player, computer
   -string name
   -char symbol
   -bool zeke
   -Space*** grid
   -Space* top, bottom, left, and right.
b. Methods:
   -Space* getTop, getBottom, getLeft, getRight
   -Fighter* getPlayer, getComputer
   -char getChar, setSymbol
   -string getName
   -bool getZeke, setZeke
   -void free, setPlayer
   -virtual turn

Kingdom class:

a. Data elements: Space elements, int food.
b. Methods: Space methods, additional methods getFood() and setFood().

The Kingdom derived space class is the space which ends the portion of the game where player traverses the map. If player arrives here, if they obtained the key from the Armory, they may give it to King Ezekiel and will fight with the Tank against Negan in the boss fight. They may also opt to give all their food to King Ezekiel and in return, he will join them in the final boss fight.

Prison class:

a. Data elements: Space elements.
b. Methods: Space methods, additional methods getFood(), setFood(), getGuns(), and setGuns.

The prison derived class allows the player to "raid" by restocking both guns and food at will, however, each time they restock, they run a chance of sustaining an attack from a walker, which they cannot defend.

Armory class:

a. Data elements: Space elements.
b. Methods: Space methods, additional methods getFood(), setFood(), getGuns(), and setGuns.

Armory derived space allows player to either heal, trade food for guns, or they can trade all their guns for a key and will no longer be able to carry guns. If they do so, they can get a key which can be traded for the tank at the Kingdom.

Hilltop class:

a. Data elements: Space elements.
b. Methods: Space methods, additional methods getFood() and setFood().

Hilltop derived space allows player to trade food to heal or to trade guns for food.

Testing plan:

While I intend to fully test every aspect of this program, the key features I would like to focus on are those outlined in the requirements. Specifically, the questions below and whether there are any issues with any of them.

Do all spaces and pointers initialize correctly? Are players correctly prohibited from going out of bounds? Does the trade/combat system implemented by the virtual turn() method work correctly for all spaces? Does the player selection for work correctly for boss battle? For example, does obtaining and giving the key to King Ezekiel create and assign the tank? Does King Ezekiel fight with you as an ally if you gift all food to him at the Kingdom? Does the game end properly once the boss battle ends?

Secondly, the questions above are primarily related to the gameplay and scenarios where the player wins. I will also need to test scenarios where the player loses. For example, time requirement (food at the Kingdom runs out) or the player dies in combat.

Finally, are there any memory leaks?

<u>Grid and bounds test:</u>
Test that all spaces are initialized correctly with no pointers set to NULL. Test bounds for each side of the board. If working correctly, correct error message will display, if not working correctly, a segmentation fault will occur when the move function attempts to access a space out of bounds.

Space tested	Error message display correctly?
Move to top space from row 0.	Yes, player redirected to move again.
Move to bottom space from row 9.	Yes, player redirected to move again.
Move to left space from column 0.	Yes, player redirected to move again.
Move to right space from column 9.	Yes, player redirected to move again.

Beyond this, I moved throughout the board systematically up and down (had to restart a few times as time limit is 25 moves), and I was able to successfully test that all spaces were initialized with all pointers pointing correctly.

<u>Trade/combat system (virtual turn method) test:</u>

Space	Turn method	Turn method	Turn method
Armory	Heal-correct	Trade-correct	Restock-correct
Hilltop	Heal-correct	Restock-correct	Leave-correct
Prison	Pass-correct	Raid-correct	N/a
Horde	Attack-correct	Hide-correct	Invisible(Michonne only)-correct.
Negansmen	Pay tribute-correct	Fight-correct	Defeated Savior become walker-correct.
Kingdom	Heal-correct	Gift-correct	Key-correct

<u>Character selections (Initial and boss battle) test:</u>

Initial character selection	Boss battle selection
Rick-correct	Select Daryl, Michonne-correct
Michonne-correct	Select Rick, Daryl-correct
Daryl-correct	Select Rick Michonne-correct
Tank- n/a	If Tank, only fighter is tank- correct
King Ezekiel- n/a	King Ezekiel join? correct

<u>Fighter status and satchel test:</u>

Fighter	Food status	Gun status	Health status	Health deducted if food < 0	Satchel (trade gun item for key item)
Rick	Correct	Correct	Correct	Correct	Correct
Daryl	Correct	Correct	Correct	Correct	Correct
Michonne	Correct	Correct	Correct	Correct	Correct

King Ezekiel, Negan, Savior, and Walker are not selectable characters so above does not apply.

<u>Game status test:</u>

Win	Lose	Game end/output message.
Player makes it to Kingdom, fights Negan.	N/a	Correct.
Player makes it to Kingdom, fights Negan with King Ezekiel	N/a	Correct
Player obtains Tank, fights Negan.	N/a	Correct.
N/a	Kingdom food supply runs out.	Correct.
N/a	Player dies on board before Kingdom.	Correct.
N/a	Player dies in boss battle.	Correct.

<u>Memory leaks test:</u>
Checked during initial incremental testing for Rick class and noticed a small memory leak due to forgetting to delete items in satchel, this was correct and final testing with valgrind on flip found no memory leaks.

<u>Reflection and changes:</u>

My final program did not deviate much from my design plan and I did not run into many issues. I initialized conceived the map of the grid as a 20 x 20 grid, however I felt this was too big of a scope for this project and would not allow time to add the additional variety of space classes I would want.

The only issue I really ran into was the initialization of the grid and spaces in the beginning. Linking all four spaces to each other required an extra step. I initially tried to initialize the pointers within the constructor, however ran into segmentation faults immediately. I essentially realized that as each space was being linked to four others, by initializing in the constructor, I was attempting to initialize a pointer to a space that had not yet been created, typically the bottom or right. To resolve this issue, in my grid setBoard method, I broke the initialization into two separate steps. I first filled the entire grid with spaces, with their respective space pointers set to null, then looped through the grid to set the space pointers to the spaces that had already been created. This immediately resolved the problem and there were no further issues.

The only other issue I ran into was a small memory leak that I realized was a result of forgetting to properly free items allocated to the satchel. This was an easy fix and I tested the application several times without any memory leaks.