The assignment is to design the Oregon Trail game using quadruply linked lists, dynamic memory allocation and freeing, and various types of structs. My code handles the game properly; however it is limited to a 1 digit map size, 1 digit port size and a maximum of 2 digit food quantity. Interfacing with the game is done through the Terminal/PuTTY. The files included in the entire game are main.c, game.c, game.h, map.c and map.h. Input .txt files are necessary to run the game. The user is able to play the game, view instructions on how to play the game and quit the game.

A few obstacles I had to overcome was handling the quadruply linked list, and freeing the maps. Getting file i/o data directly into a quadruply linked list required many variables for storing various key points for creating the linked list. For example, I required variables that stored the adjacent linked list item so that I could link up & down relationships in my linked list. I also had issues handling the ocean. I figured out that if I recognized after what point the ocean exists, I can just create ocean after that specific point, in the form of the quadruply linked list.
A note I would like to make about my project is that I had not realized until too late that the viewports were to be allocated memory space. I thought that the map had to be dynamically allocated in memory. I understand my mistake in this matter, so I tried to show that I am capable of applying the concepts of utilizing malloc() and free() properly when applying it to map. The only difference is that I do not call it multiple times. I do not implement it due to the fact that I already had a working system for the game. The DeleteMap() function will be able to delete any sized map; however, for some reason, it consistently fails to free one, and only one, allocated memory block.

The main functionality of the code is playing the game. The game algorithm is as follows:

1. Generate the map using dynamic memory allocation
2. Display the viewport to the user
3. Ask the user to move up, down, left or right and, depending on what is on the tile, the user will interact with the space.
    a) If the tile was an animal: ask to hunt or retreat. If they hunt it, their amount of food changes depending on the type of animal, the animal dies and nothing resides in the space, and they progress into the space. If they retreat, they know what is in the space, but remain in the one they were in.
    b) If the tile was a disease, automatically contract the disease. They lose a certain amount of food depending on the disease, and also move into the space.
    c) If the tile is a river, the user has the option to cross or to retreat from the river. If they decide to cross, they will move into the space, but lose a specified amount of food. If they retreat, they remain in the space they were in. Rivers are always viewable.
    d) If the tile is an ocean, they are not allowed to move there.
4. Continue to repeat step 3 until food supply is exhausted, which triggers a losing screen, or if they reach their destination, which triggers a winning screen.

The instructions that are shown to the user, upon request, is shown below:

```
------------------------INSTRUCTIONS--------------------------
Game Description: You are traversing through a map and have
     limited food resources. Encountering various obstacles
     will use resources. Your goal is to reach the destination
     before you exhaust your food resources.
Controls: You will type the following characters to control
     your movements:
       up - 'u'   down - 'd'   left - 'l'   right - 'r'
View Port: You will only be able to see a portion of the map
     through the view port that will be shown with each
     move made. Various letters will represent what is at the
     space. Arriving at a space will reveal what is there.
     X - Your current location
     Z - Your destination
     U - Undiscovered area
     O - Ocean which borders the map. You cannot go here.
Obstacles: You will encounter obstacles in the form of
     rivers, animals or diseases.
Rivers: You will be given the option to retreat from or cross
     the river.
                R - River       ->  -20 Food
Animals: You will be given the option to retreat from or hunt
      the animal. Hunting results in various food alterations.
                G - Grizzly     ->  -10 Food
                B - Boar        ->  - 5 Food
                E - Elk         ->    0 Food
                H - Hare        ->  + 5 Food
Diseases: Once you step into a disease space, you contract
      the disease automatically.
                C - Cold        ->  - 5 Food
                F - Flu         ->  -10 Food
                D - Dysentery   ->  -15 Food
Winning Condition: You win if you reach the destination before
      running out of food.
Losing Condition: You lose if you run out of food before
      reaching your destination.
------------------------------------------------------------
```

To run the game, there are three options:

1. `./proj3 <inputfile.txt>`
2. `make run FILENAME=<inputfile.txt>`
3. `make val FILENAME=<inputfile.txt>`

This project has taught me about the value in dynamically allocating memory and that various types of structs are useful for different tasks. I also learned about how linked lists can become very complex when adding multiple links.