Final Project Document
Author: Glenn Frutiz
Date:8/13/19

What the user will see

Welcome to the Dog Walking Simulator. The purpose of this game is to walk your dog until they poop then walk back home. The catch is that you own a very jumpy dog that will pull the leash at any sort of stimuli. Everytime the dog pulls on the leash it will be damaged. To stop the dog from pulling the leash you can use a treat to grab its attention.

1. Play Game
2. Exit Game

If Play Game is chosen	You have chosen to exit the game.
You have chosen to play the game. [Home] - [ ]- [ ] - [ ] - [ ] You are currently home, enter a direction you would like to go. 1. Left 2. Right Regardless of the direction the player chooses. [Home] - [dog]-[ ] - [ ] - [ ] (Implement Random Event Generator)	Thank you for playing!

Your dog was a good boy on this space	Your dog saw a squirrel and lunged at it. The leash was damaged.	You found a treat!

Leash Strength Points: (points will be

displayed) (increase steps taken) enter a direction you would like to go. 1. Left 2. Right	

(If the dog has not pooped yet) You can't go home if the dog has not pooped yet	(If the dog has pooped)
You can't go home if the dog has not pooped yet Enter a direction you would like to go. 1. Left 2. Right	Welcome Home! Final Leash Strength Points: Thank you for Playing!

Plan

I will adapt my work from Lab 6 and Lab7 for this project.

My idea for my game is a dog walking simulator. Outside of working on my assignments I have been spending a lot of my time walking my two dogs. They are prone to pulling on the leash, so I will make the game have a health system for the leash. The object of the game is to walk a certain amount of steps to get the dog to poop and walk back home without the leash breaking.

Space class/Node class
I will take my work from Lab 6 to help me make the space class. I will have the Node class I made function as the Space class. I will maintain the previous and next pointers, but I will add right, left, up, and down pointers as well. These pointers will be initialized depending on where the space is. I will also add a boolean value as a member variable to indicate whether there is a

dog on the space. There will also be a string member variable that will be used for displaying the space.
The derived classes will be LeashPull Space, Enemy Dog Space, Treat Space, and Good Dog Space.
Space class

1. Member Variables

    a. Space* top
    b. Space* bottom
    c. Space* left
    d. Space* right
    e. std::string type
    f. Std::string prompt
2. Functions
    a. std::string::getPrompt()//pure virtual function
    b. virtual void modifyLeash(&int leash) = 0//pure virtual function;
    c. Std::string getType()//returns type that is initialized in constructor;

Derived Space Good Dog

1. Member Variable
2. Functions

```
GoodDog()
{
  type = “Good Dog”
  prompt = “Your dog was a good dog and didn’t pull on the leash on this space.”
}
Virtual void modifyLeash(&int leash)
{
  leash = leash
}
```

Derived Space LeashPull

3. Member Variable

    int leashDamage
4. Functions

```
LeashPull
{


}
```

Derived Space getItem

5. Member Variable
6. Functions

Derived Space Home

7. Member Variable
8. Functions

Derived Space Poop

9. Member Variable
10. Functions

Derived Space

11. Member Variable
12. Functions

Container
There will be a container for dog treats. There will be a limit of 4 dog treats in the container. The will be used to help prevent the dog from pulling on the leash.
I will use two queue member variables. Queue is a first in first out structure. I did not use a stack because I felt that it would achieve the same purpose.

Bag class

1. Member variables
    a. queue treatPocket
    b. queue baggiePocket
2. Functions
    a. pushTreatPocket
    b. pushBaggiePocket
    c. popBaggiePocket
    d. popTreatPocket

Item class
What will the treats be for? If a player chooses to use a treat, it will be popped from the queue, it will cause the leash health to not be decreased.
A baggie will set the poop bool variable to true in the organizing class.
A treat will set the treat used member variable to true in the organizing class.
At the end of each turn the treat used member variable has to be set to false.
These member variables will be used in functions

Treat derived class

Baggie derived class

Organizing Class
Similar to how I made Lab6 I will make a class that has doubly linked list as member variable. The class will also keep track of the health bar, and it will have a container for items such as a poop bag or treats.
There has to be a function that calculates damage every turn and then applies the damage.

It uses a bool member variable to determine whether damage will be calculated.
There has to be a random number of steps to take before your dog poops. For now I will only have the dog poop once.

Notes from development
I was able to make my code from Lab6 make a doubly linked list with 10 spaces. I will attempt to make the doubly linked list circular later on today.
I modified several functions from lab7. The first was the addN2Tail() function. This function creates a node in the list if there none or if nodes already exist it creates a node at the tail of the list. I changed the function by having the next pointer of the tail point to the head.
I also modified the printForward function. When I tried to use it with a circular linked list I was stuck in an infinite loop. I fixed this by setting the next pointer of the tail to nullptr, printing the contents of the list, and setting the next pointer of the tail back to the head.
To make sure I freed memory without being stuck in an infinite loop I applied the same approach to my deallocateList() function.

I will take my node class and make it an abstract base class so that it can be the space class. The node class will have a member variable for a description of the type of space. It will have a member variable for a prompt when the dog lands on it.
I will add top, right, left, and bottom pointers to the Node class. Depending on the situation I will set them equal to the previous and next pointers.

To do list

1. Set up a circular doubly linked list
    a. I worked on this on wednesday
       I modified several functions from lab7. The first was the addN2Tail() function. This function creates a node in the list if there none or if nodes already exist it creates a node at the tail of the list. I changed the function by having the next pointer of the tail point to the head.
       I also modified the printForward function. When I tried to use it with a circular linked list I was stuck in an infinite loop. I fixed this by setting the next pointer of the tail to nullptr, printing the contents of the list, and setting the next pointer of the tail back to the head.
       To make sure I freed memory without being stuck in an infinite loop I applied the same approach to my deallocateList() function.

2. Be able to deallocate a circular doubly linked list
    a. I worked on this on wednesday
       To make sure I freed memory without being stuck in an infinite loop I applied the same approach to my deallocateList() function

3. Display that I am accessing a node in a doubly linked list
    a. I worked on this on saturday
       I disabled the changes I made on Wednesday to get this to work
       I added another pointer to the DLL class named current.
       I added functions to setCurrent, and to get current

4. Be able to get to the next node and the previous node in a doubly linked list
    a. I made setCurrent have a menu which asks users whether they would like to get to the next node or previous node.
       It validates that the node the user will access exits otherwise it will ask the user for another choice.
5. Display that I am accessing a node in circular doubly linked list
    a. I made a function that helps me get the value stored at the current node.
6. Be able to get to the next node and previous node in a circular doubly linked list
    a. I just called the function setCurrent and since it was a circular doubly linked list i didn't have to worry about any Next or Prev pointers being null
7. Now that I can get from one place to another I will add string member variables to the nodes to give the appearence of a board. I named the member variable space
    a. I added functions to the Node class to get the value of the string and to set the value of the string.
    b. I added the getSpace function to the printForward function and made the function display the board.

<u>Notes During Development</u>

8/10/19
I was able to make a circular linked list that was initialized with different types of spaces. These spaces are derived classes of the Node class. I implemented a mechanic for moving from space to space. This was done by adding a pointer to the doubly linked list class and changing the contents of the pointer after asking the user for their choice. I also implemented a mechanic that randomized the type of space except for the starting space which was set as home. I wanted to set a mechanic that rerandomized each space but I struggled with deleting a node that is not at the head or tail. To save myself time I cut my losses and continued programming. Afterwards I sought to implement the data members that would keep track of the leash strength. However I found myself debating whether I should implement this in the DLL class or in the Menu class.

8/11/19
I made derived classes of the Node class. These classes ended up being Home, GoodDog, TreatFound, and LeashPull.

In the node class I made a pure virtual function that displays the event necessary for each class. It is a void function to display the message unique to each space.

- The Home class is a goal for the player to reach after walking the dog a certain amount of steps.
- The GoodDog class is a class that doesn't cause the Leash Strength to decrease. Kind of like a free space.
- The TreatFound class causes the queue that holds the treats to increase.
- The LeashPull class causes the Leash Strength member variable to decrease. It will also ask the user whether they want to use a treat. If a treat is used then the leash strength doesn't decrease

I was stumped on how to make the spaces communicate with the variables that track Leash Strength and how many spaces the dog walks.
I realized that all of my nodes only displayed messages saying what would happen on that node. I changed this by making the functions for event return integers.

- I then made a switch statement that would take the different integers returned by the derived classes.
- Depending on the integer taken it would perform the different events for each space.

I kept getting valgrind errors yet no memory leaks. Based on my previous assignments I solved this by making sure I didn't leave any hanging pointers.
I then set up my menu class.
I tried to make a container for treats inside DLL but I couldn't make it work. I kept getting valgrind errors I will end up making the container outside of DLL.:
8/12/19
After I made the container outside of DLL it worked properly.
I will take things a set further and modularize more.
I will make a class for the dog as well. I had DLL doing all of the heavy lifting. I kept getting valgrind errors when I tried to decrease leash strength. Instead of making a class for the queue I just made a queue in DLL this stopped valgrind errors from occurring. I think It was because I was allocating more memory inside of memory that was already allocated.
I moved the switch statement from the DLL class to the Menu class as well.
Now the DLL class only organizes the different spaces and randomizes them when the doubly linked list is made. The Dog class organizes the leash strength and whether the dog has pooped. The Menu class keeps track of whether the game starts or finishes.
8/13/19
I tested the program running valgrind to make sure there would be no issues. I changed the name of the game to That Dang Dog! and added some more cute and humorous lines. I added calls to std::endl to space out some of the output of the program.

Purpose of Test	Inputs	Expected Outcome	Actual Outcome
See if the user can input anything other than 1 or 2 into Menu that is displayed at the beginning of the game.	Test1: 0 Test2: 00 Test3: -1 Test4: 1.0 Test5: 4 Test6:1000000000000 Test7: $%^@# Test8:asdf Test9:2.0 Test10:1.00000000000 Test11:2.00000000000 Test12:1asdf Test13:2asdf	Number must be an integer in range 1 to 2.	Number must be an integer in range 1 to 2.
See if the user can input anything other than 1, 2, or 3 into Menu that asks whether they would liked to Left, Right, or Exit Current Game.	Test14: 0 Test15: 00 Test16: -1 Test17: 1.0 Test18: 4 Test19:1000000000000 Test20: $%^@# Test21:asdf Test22:2.0 Test23:1.00000000000 Test24:2.00000000000 Test25:1asdf Test26:2asdf	Number must be in integer in range 1 to 3.	Number must be in integer in range 1 to 3.
See if I can go through Doubly Linked Node by only pressing the previous Node.	Test27: Select the Previous Node until the leash breaks Test28: or until User returns home after the dog has pooped.	The game ends and shows the start menu.	The game ends and shows the start menu.
See if I can go through Doubly	Test29: Select the Previous Node until	The game ends and shows the start	The game ends and shows the start

Linked Node by only pressing the Node.	the leash breaks Test30: or until User returns home after the dog has pooped.	menu.	menu.
See if I can exit the current game.	Test31: Select option to return to current game	The game ends and shows the start menu.	The game ends and shows the start menu.
See if I can enter something other than 1 or 2 when I am asked to use a treat	Test32: 0 Test33: 00 Test34: -1 Test35: 1.0 Test36: 4 Test37:1000000000000 Test38: $%^@# Test39:asdf Test40:2.0 Test41:1.00000000000 Test42:2.00000000000 Test43:1asdf Test44:2asdf	Number must be integer in range 1 to 2.	Number must be integer in range 1 to 2.
See if game crashes If I choose to use a treat.	Test45:Select option for using a treat.	You used a treat your dog did not pull on the leash.	You used a treat your dog did not pull on the leash.
See if game crashes if a choose to not use a treat	Test46:Select option for not using a treat	You chose to not use a treat The LeashStrength was decreased	You chose to not use a treat The LeashStrength was decreased
See if game crashes if the leash breaks	Test47:Play until the leash strength is at 0	Start menu appears	Start menu appears
See if the game crashes if the user returns home after the dog has pooped.	Test48:Play until the dog poops and then return home.	Start menu appears	Start menu appears
See if the game ends	Test 49: The game	You are home. You	You are home. You

when I return home without the dog having pooped	does not display the start menu	can’t go home if the dog hasn’t pooped.	can’t go home if the dog hasn’t pooped.

Reflection

Compared to the beginning of the quarter my skills have improved. I feel confident that I know how to use a lot of syntax and concepts properly. If I come across a situation where I don’t feel confident I know where to look for help.

After doing this project some areas where I can still improve include knowing where to modularize. After the Zoo Tycoon project I took modularization to the extreme, yet during Project 3 I felt as if I had barely modularized at all.

During this project if I had barely modularized I kept getting memory leaks. I made my doubly linked list class (DLL) keep track of the spaces and their derived classes, and keep track of the Leash Strength along with any associated data members and functions. I kept getting valgrind errors saying that I was trying to write data at invalid addresses. I managed to fix this error by dynamically allocating the instance of the DLL class in the Menu class. The straw that broke the camel’s back was adding a queue type container to the DLL class. No matter what I did, I couldn’t get it to work.

I thought my code seemed very cluttered and confusing. Based on the reports from valgrind I also couldn’t tell where errors were coming from. I modularized by making the Dog class, which tracked whether the dog had pooped and leash strength. I only had the DLL class track the nodes. Also I had the Menu class track what kind of space the user was on and whether the game had been won or lost.

I think my problems are caused by my relative inexperience designing projects. Before the start of the next quarter I will continue to practice my skills by making another project. I will attempt to make a seating chart that generates a random seating arrangement for a list of students. It will take into account whether pairs of students should sit with each other or not. I tried to make this project before CS162 started but I had no idea how to approach it. Now that I know more about inheritance, classes, containers, I feel that I will be able to implement this project. Most importantly, I will be able to practice my skills for designing projects.