Austin Henry CS 162 Summer 2018 Final Project 8/15/2018

Program Idea:

Pokemon themed game where you must get a gym badge in order to prove to Professor Oak that you are ready to become a Pokemon Trainer. The game will consist of traversing between Pallet town and Pewter City. Because pokemon is an RPG game, the key to success here is gaining levels in order to defeat Brock. There is not a puzzle element to this program. Instead you must gain enough levels and beat Brock to acquire the Boulder badge. The setting includes a time limit of 30 days to complete this task, otherwise your mother and Professor Oak will determine that you are not ready to begin a Pokemon Journey.

-Levels can be gained by beating a wild pokemon or by using a rare candy
-You may only have 1 rare candy
-Traveling north or South on a **route space** gives you a ⅓ chance of gaining a rare candy
-**route spaces** will have you battle pidgeys if you select to battle a wild pokemon
-**viridian forest space** will have you battle a caterpie if you select to battle a wild pokemon
-**pallet town space** allows you to talk to professor oak
-pewter city space allows you to use a pokemon center to heal, and also fight brock
-Because Pallet Town to Pewter City is linear in the Pokemon games, it is also linear in this demo version

You achieve victory by beating Brock to get the Boulder Badge, and then traveling all the way back to Pallet Town to show Professor Oak the badge. If you do not select “Talk to Professor Oak” once returning to Pallet Town, you will not have shown him and thus will not achieve victory.

Game Layout is:

Room 1 →	Room2 →	Room3 →	Room4 →	Room5 →	Room6
Pallet Town	Route 1	ViridianForest	Route 2	Route 3	Pewter City

Trainer Class

Private:

```
Space* currentSpace;
Monster* currentPokemon;
Vector <string> items
```

Public:

```
Void initializePointer //Puts the player on the first space
Space* getSpace   //Returns the space the player is on
Void moveNorth; //Moves the player North
Void moveSouth; //Moves the player south
Void checkPokemon (&Pokemon) //Displays Pokemon Stats
```

Void healPokemon //Heals the pokemon
Bool checkbadge //Checks to see if player has boulder badge in items
Void giveBadge //Gives the player a badge item
Void giveRareCandy //gives the player a rare candy
Void useCandy //Gives the player a rare candy

Space Class

Protected variables

Space* north;
Space* south;
Space* west; //will be null
Space* east; //will be null

Abstract portion: (virtual function = 0)

Virtual Void print = 0 //Prints out picture of the map in characters
Virtual int menuDisplay(Trainer*) =0; //Prints out menu, activates any special events

Non-Abstract portion

Void setPointer(*Space north, *Space south) //Sets pointers
String getAreaName //returns the name of the space
space* getNorth //returns the north space
space* getSouth //Returns the south space

Pallet Town Class

Menu Options:

1. Move North, 2. Move South, 3. Talk to Prof Oak, 4. Check Pokemon, 5. Use candy, 6. Exit program

Functions:

Void talkToOak //Lets you speak with Professor Oak (checks for special item)
Menu (Trainer*) //Displays the menu with Pallet town options

Route Class / Viridian Forest Class

Menu Options:

1Move North, 2Move South, 3 Check Pokemon Statues, 4 Battle Wild Pokemon, 5 Exit

Functions:

menu(Trainer*) menu, gives random chance to gain rare candies if it is a route class

Pewter City Class

Menu Options

1Move North, 2Move South, 3Check Pokemon, 4Go to PokeCenter 5 Use candies, 6Battle Brock 7. exit

Functions:

Void pokemonCenter(&Pokemon)

Monster Class (Parent) //Same as ratata, geodude, onix, caterpie

Protected:

Int defense;

Int strength;

Int health;

Int initalhealth;

Int level;

String name;

Public:

Monster(defense, strength, health, level, name)

Void levelUp //For leveling up charmander to increase stats

Virtual int attack() = 0 //Redefined attacks for each pokemon

Void defend(int) //Defends against an attack parameter

Bool isDead //If the pokemon is dead

String getName //Returns name

Void healMonster //Heals the pokemon

Void rest //For when the player checks on the pokemon

Void rareCandy //Gives a special level up when a rare candy is applied

Charmander/Squirtle/Bulbasaur Class

New functions:

Void levelup

Main Function

```
//Allocate the spaces
Space* room1 = new PalletTown()
Space*room2 = new Route1;
….. etc..
Space*room6 = new PewterCity()

//Linking the spaces
room1->setPointer (room2, nullptr)
room2->setPointer(room3,room1)
...etc…
room6->setPointer(nullptr,room5)
```

```
//Initializing the trainers Pokemon as well as the trainer
Monster* m1 = new Charmander(stats here);
Trainer player1(room1, m1) //Gives starting room and starting pokemon
Trainer* playerPointer = &player1 //Easier to pass trainer as a pointer


/*******************GAME***************/
/*
Game relies on the menu inputs of the user. This is stored in the variable called menuDecision
0 = no response
1 = move north
2 = move south
3 = wild battle with pidgey
4 = battle with brock
5 = battle with caterpie

9 = exit the game
*/

Cout statements introducing game.
Int counter = 0

//This line prints the menu of the space the player is on and it passes the player as a
//parameter for any special events that may occur
Int menuDecision = playerPointer->getSpace()->menuDisplay(playerPointer);

While (menuDecision !=9 && counter !=30)
{
        If (1)
                moveNorth
        if(2)
                moveSouth
        if(3)
                Implement battle with pidgey
                -allocate pidgey
                -check to see if charmander is dead
                -commence battle
                -Display pokemon and their hp/levels
                -Allow user to pick to fight or run away. If the user runs away end the battle
                -If user attacks, display each pokemons new hp again
                -Automatically make pidgey attack
```

-Keep going until one pokemon dies
-If pidgey dies, give charmander a level up
-delete pidgey at end of battle whether user won/lost/ran away

If (4)

Battle with Brock
-Uses same system as battle with Pidgey
-This battle however has two Pokemon. Brock has a geodude and an onix
-After user beats geodude, have it battle onix
-If user survives both battles, give the boulder badge item
-delete both onix and geodude if user battles both. Delete just geodude if user loses to geodude and never battles onix

if(5)

Battle with caterpie
-is exactly the same as battle with pidgey mentioned above

If (9)
{
-Exit the game
}

//After all if statements are executed.
Cout days left
menuDecision = playerPointer->getSpace->menuDisplay(PlayerPointer)
Increment days
}

//end of while loop

//delete all allocated rooms and monsters/trainers

Testing table:

Test Case	Input Values	Tested Function	Expected Outcome	Observed Outcome
Input too low	Numbers < 1 i.e. (0, -1…)	Input validation for all menu Decisions	For validation: “Please input a number in between min and max	Same as the expected outcome
Input in correct range	Min < numbers < max	Input validation for all menu Decisions	Same as above	Same as the expected outcome
Incorrect data type input (char or string, when int required)	Abc, a, a a a , 1a1a, ‘ ‘	Input validation for all integers assigned by input	Same as above	Same as the expected outcome
Destruction and De-allocation for Monster, Trainer, and Space allocated with new	When the delete lines are called at end of program. As well as delete lines during battle	Arrays allocated by new	Delete functions are successfully called at the correct time	Same as the expected outcome. Valgrind shows no leaks

Item successfully added to vector (rare candy, boulderbadge)	Call functions to add boulder badge and rarecandy	Void givebadge Void giveCandy	Adds an item to the vector as expected. Does not exceed limit of item	Same as expected outcome
Menu function have logical pathways	Menu functions within space classes	Menu within pallet town, route, viridian forest, pewter city	Successfully runs all menu choices without any errors or undesired outputs	Same as expected outcome
Trainer is not allowed to fight/continue fighting if charmander dies	Charmander vs pidgey, vs caterpie, vs geodude, vs onix	Battle sections within main function	Successfully prevents user from battling with a fainted pokemon	Same as expected outcome

Reflection:

I found that this was my favorite assignment during CS162. Not only was I able to successfully program a “demo version” of a game I love, but I was able to create an entire system by myself. My goal upon entering CS162 was to be able to become a more independent programmer, and I feel that I have met this goal.

While programming this, I at first encountered a large number of issues because I had been using circular inclusion without my knowledge. This led to a number of design changes that eventually landed on me setting up a trainer pointer and passing it to my menu functions. This meant that I would be free to implement my battle systems within main as well as execute any special events having to do with the trainer within my space classes. This proved to be very beneficial to the overall design and ease of understanding my program. By having the battles be “pre-setup” within main it allowed me a lot of freedom when it came to memory allocation. This

made it simple to allocate and deallocate the enemy pokemons within main so that I did not have to chase them down and add delete statements to one of my many many classes.

By and large the hardest part of designing and implementing this assignment was deciding the scale of my project. In the end I scaled way back on my initial goals. I had plans to allow the trainer to carry multiple pokemon, to catch pokemon, and to deposit pokemon into a PC (like in the games). This resulted in remnants in my code that would work for multiple pokemon, even though the user will only ever use Charmander. I also had planned to print out a text version of the map, which I decided would take to long. Lastly, I planned on implementing an entire market system to purchase and sell items allowing the user some additional freedom. I felt that I needed to scale all of these aspects way back if I wanted to finish this assignment in time. I believe that in the future I will add these aspects, for now I will keep them as ideas.