Taylor Buchheit
CS162 – Final Project
6/4/2014

Requirements

- I will modify my creature fighting program so that it is a tournament style fighting program.
- The first player will choose a list of Creatures, then the second player will choose a list of Creatures.
- The players must choose the same amount of creatures.
- I will need to keep track of the standings, ie, keep track of the Creatures who finish first, second, and third in the competition.
- After a fight, the winner goes to the end of the list of Creatures on its team, the loser goes into the list of dead creatures.
- I can decide if the Creatures will heal after their fights they have won.
- I must use two different data structures in the last two chapters of our reading.
- I submit a short note that checks with Professor Rooker or a TA if my chosen data structures are optimal for the assignment.
- I need to test my program with all types of Creatures types I have created.
- At the end of the tournament I should print the places of the Creatures and which player has won the tournament.
- I'll need to research different standings systems and clearly explain which one I am using in the design document.

Assumptions

- I assume I will use the program from Assignment 4 to start with.

Design

I have asked Professor Rooker if using a Stack for the dead Creatures and using a Queue for the alive creatures would be good data structures. I think a stack would be good for the dead Creatures because the dead Creatures will go into the stack in order, and can be taken out of the stack in reverse order. This is perfect for printing the Creatures places because the last place Creature will have died first. The queue would be good for alive Creatures because it goes in order, and when a Creature wins a round, it will be placed at the end of the Queue. If it dies, it goes in the stack of dead Creatures. Professor Rooker agreed that these are good choices.

There isn't too much I think I will need to design since the layout of the program is pretty similar to the previous assignments. I will need to create a Stack class that can hold Creatures, and I will need a Queue class that will hold Creatures and Weapons. When the program starts, the program will ask the users how many Creatures they would like to choose. That number is then used to determine the size of the Stack and Queues. Once they have decided the amount of Creatures they want, player 1 will get to choose the Creatures they want to fight with, and equip them with their Weapons. A queue for Player 1 will be created before Player 1 chooses their first Creature. Then Player 2 will choose their Creatures they want to fight with and equip them with Weapons. Another Queue will be created before Player 2 chooses their Creatures.

Once the Creatures and Weapons have been chosen for both Players, the first duel will take place. If the Creature wins, it will be popped off of the Queue and added to the end the Queue. If the Creature loses, it will be popped off of the Queue and placed in the Stack.

When all of one team's Creatures are dead, there will be an analysis that determines the standings of Creatures, and the winner of the Tournament. Since there will be an equal number of Creatures on both teams to start, every Creature must fight at least once. This means that every Creature has a chance to fight. I like the thought of doing a standing system that is essentially a last man standing style. The last three Creatures to die will place the highest. If there are three Creatures left on a team, those three Creature's wins will be compared to one another. If they are all the same, they will all tie for first place. It could be possible that two Creatures tie for first place, or for third place. Otherwise, they will all just place first, second, or third.

I have decided that if a creature wins a round, they should heal for winning as a reward for their efforts. When a Creature wins they will be healed for a flat 3 strength. This means if the Creature did not receive any damage from the last fight, they will start with 3 more strength than they normally do. However, if they have been damaged and only have say 2 strength left, they will gain 3 more strength and be brought up to 5 total strength for their next round of fighting.

Functions needed in main

FillQueue()	Will be used to create the Creatures and at them to the queue for each team.
FillStack()	Will be used to fill up the stack of dead Creatures. Will take the dead Creature off of the appropriate queue and put it on the Stack.
PutBack()	Will be used to take the winner of the current fight off the front of the appropriate queue and put it at the end of the queue.
PrintQueue()	Prints the Creatures in the queue.
PrintStack()	Prints the Creatures in the stack in reverse order.
DetermineWinner()	Will be used to determine which team wins. If the team 1 queue has creatures at the end of fighting, it will print that team 1 wins. If the team 2 queue has creatures at the end of fighting it will print that team 2 wins.
Heal()	Will be used to add the 3 points of strength to the winner of the current fight.
FreeMemory()	Will be used to delete the queues and stacks at the end of the program to ensure there are no memory leaks.

I think this should cover the functions I need to add. I can foresee adding a few variables to the Creature class such as int wins to hold the number of wins each Creature has, and a string team that holds a string of which team the Creature is on. Beyond this, there shouldn't be much I need to modify in the other classes. The main area of programming will take place in main since the structure of the program is going to drastically change. I will probably need a while loop that determines of either of the queues are empty to know if the tournament has ended. I will also need a while loop inside of this while loop that carries out the fighting, like I did in the previous versions of this program. I think I will design it so that the weapons are chosen at the beginning of each round of fighting since the Queues will only hold Creatures. Then the users have a little more interaction during the fighting process, and adjust their strategy for what weapon to choose based on who they are fighting, and how they are doing in the tournament. I don't see much of a need for editing the functions I currently have set up for fighting, but I may find a need to change these when I start programming.

Overall, I think I have a decent idea of what I need to do in order to turn the program from Assignment 4 into a tournament style program.

Testing

For testing this program, I will go through a program that uses 3, 4, 5, and up to 12 Creatures per team to ensure that it is working properly. I will use a mixed set of Creatures so that we can have as different array of outcomes as possible.

Number of Creatures	Team 1 Creatures	Team 2 Creatures	Winning Team	Top 3 Creatures	Oddities
3	Goblin, Barbarian, Reptile	Blue Man, Reptile, Barbarian	Team 2	**First:** Team 2 Blue Man, **Third:** Team 1 Reptile, **Third:** Team 1 Barbarian	None, both teams lost members and the outputs displayed correctly
4	Reptile, Barbarian, Blue Man, Goblin	Barbarian, Reptile, Barbarian, Blue Man	Team 2	**First:** Team 1, Blue Man **Second:** Team 2, Reptile **Third:** Team 2, Blue Man	Even though the first place Creature was on Team 1, it said that Team 2 won. This help me discovered that I had called the determineWinner function in the wrong place in main. I moved it to before the rest of the stack is emptied.
5	Barbarian, Reptile, Barbarian, Blue Man, Goblin	Blue Man, Blue Man, Reptile, Goblin, Goblin	Team 1	**First:** Team 1, Blue Man, Team 2 Blue Man **Third:** Team 1Goblin	I'm not sure how Team 1 Blue Man and Team 2 Blue Man tied for first since Team 1 Blue Man killed Team 2 Blue Man. I guess if they have the same amount of wins this is possible, I hadn't thought of that. I added a check in

					the printStack to check to ensure the Creatures have the same team if they tie for first. This should fix the problem
6	Blue Man, Blue Man, Reptile, Barbarian, Reptile, Goblin	Barbarian, Barbarian, Reptile, Goblin, Blue Man, Goblin	Team 1	**First:** Blue Man2, and Blue Man1 **Third:** Goblin	The only thing I notice that is odd is that the Barbarian and the Goblin survived and should have the same number of wins buy the Goblin placed higher than the Barbarian. This may have to be an addition I add later because I am not sure how to deal with it.
7	Blue Man, Barbarian, Barbarian, Goblin, Reptile, Goblin, Goblin	Goblin, Reptile, Barbarian, Reptile, Reptile, Reptile, Blue Man	Team 1	**First:** Team 1, Blue Man1 **Second:** Team 2, Reptile 6 **Third:** Team 2, Blue Man 7	No problems noticed with this run, although I have been noticing that none of the Creatures have been becoming plagued. I looked back at the code and realized that my if statements in my plaguedCreature function check for the name instead of the choice. I changed this to look at the choice and think I have fixed the problem.
8	Blue Man, Barbarian, Reptile, Goblin,	Blue Man, Plagued Barbarian, Reptile,	Team 2	**First:** Team 2, Blue Man 7, Team 2, Reptile 3, Team 2 Blue Man 1	No problems, and we can see that the plagued function is now working

	Blue Man, Reptile, Barbarian, Goblin	Goblin, Plagued Blue Man, Barbarian, Blue Man, Blue Man			again.
9	Plagued Reptile, Plagued Blue Man, Plagued Reptile, Barbarian, Blue Man, Plagued Goblin, Barbarian, Plagued Goblin, Goblin	Goblin, Barbarian, Barbarian, Plagued Blue Man, Goblin, Blue Man, Plagued Reptile, Barbarian, Reptile	Team 1	**First:** Team 1, Blue Man 5 **Second:** Team 2, Plagued Blue Man 4 **Third:** Team 2, Reptile 9	This was actually exciting to watch because the Team 1, Blue Man 5 was the only remaining Creature against 4 on Team 2 and it came back to win. Maybe the heal helped him win.
10	Reptile, Barbarian, Reptile, Reptile, Plagued Reptile, Goblin, Blue Man, Goblin, Barbarian, Reptile	Plagued Goblin, Plagued Goblin, Barbarian, Goblin, Plagued Reptile, Goblin, Reptile, Barbarian, Plagued Reptile, Goblin	Team 1	**First:** Team 1, Goblin 6, Team 1, Reptile 4, Team 1, Reptile 1	Nothing out of the ordinary, other than again having the issue when more than three Creatures remain on the winning team and displaying only the last three on the stack as being in first place.
11	Goblin, Reptile, Barbarian, Blue Man, Blue Man, Plagued Barbarian, Reptile, Plagued Barbarian, Goblin, Plagued Goblin, Plagued Barbarian	Plagued Blue Man, Barbarian, Plagued Goblin, Plagued Blue Man, Plagued Barbarian, Plagued Blue Man, Blue Man, Blue Man, Reptile, Blue Man,	Team 2	**First:** Team 2, Barbarian 11, Team 2, Blue Man, Team 2, Reptile 9,	Right at the end of running this round, I got a segmentation fault. I probably have a memory leak in the program somewhere. I wonder if this happens with bigger number of Creatures, or if its because I have run the program so many times

		Barbarian			without relaunching it. I'll see what happens with my run with a 12 Creature line up and see if it does it the same.
12	Goblin, Barbarian, Reptile, Plagued Goblin, Blue Man, Barbarian, Reptile, Blue Man, Barbarian, Reptile, Goblin, Blue Man	Plagued Blue Man, Barbarian, Reptile, Goblin, Reptile, Blue Man, Barbarian, Blue Man, Reptile, Barbarian, Goblin, Reptile	Team 1	**First:** Team 1, Blue Man 12, Team 1, Blue Man 5 **Third:** Team 2, Blue Man 6	This ran without problems. I wonder why there was a segfault on the previous run through? I'll try again with 11 and see what happens.
11	Reptile, Barbarian, Blue Man, Goblin, Reptile, Barbarian, Plagued Blue Man, Goblin, Plagued Barbarian, Reptile, Plagued Goblin	Barbarian, Goblin, Reptile, Blue Man, Barbarian, Blue Man, Barbarian, Barbarian, Goblin, Barbarian, Reptile	Team 2	**First:** Team 2, Blue Man 4 **Second:** Team 1, Blue Man 3 **Third:** Team 1, Reptile 5	No problems here. I wonder what caused that. I will try running one more time with 11 and not closing the program to see what happens.
11	Goblin, Reptile, Plagued Barbarian, Blue Man, Plagued Barbarian, Reptile, Blue Man, Goblin, Blue Man, Barbarian, Goblin	Reptile, Barbarian, Plagued Reptile, Reptile, Reptile, Reptile, Goblin, Barbarian, Blue Man, Plagued Barbarian, Plagued Goblin	Team 1	**First:** Team 1, Blue Man 7 **Third:** Team 2, Barbarian 8, Team 2, Reptile 5	It worked again according to plan. This was another fun one since Blue Man 7 on Team 1 made a huge come back. I'm not sure what caused the Segfault, but I can't seem to reproduce it.

After a lot of testing, I think the program works pretty well for the most part. One thing I wish I could do to improve the readability is make a clear screen. This always seems to be a tricky thing for me working cross platform. I could try a system("CLS"); but that only works on windows as far as I'm aware, and most people advise against it. Overall though, I think the program works the way I had intended. There are a few areas I can work to improve, like when there are more than 3 Creatures on the same team living after their team wins, it should provide the proper places for all of the Creatures and not just use the last three on the stack. I'm a bit baffled as to why I got the segmentation fault one time and not any other times. Perhaps its because I had run the program through so many tournaments without closing and reopening. I'm not quite sure because I tried to reproduce it and couldn't. I have a function that clears the queues, so I'm not sure where the memory leak would be. After you pop something off the queue or stack, are you supposed to delete that memory that was holding the object being popped? I thought this was taken care of for the programmer, but I could be wrong. Hopefully there are no memory leaks, but there very well could be and I'm not sure how to test other than to run the program many times like I just did for the last hour and a half.

Reflection

I really enjoyed this project, along with the other adaptations to our Creature fighter assignments. I was surprised at how easily I felt I was able to adapt the program from Assignment 4 into the Final Project. Using the STL made the program so much easier to deal with since they had preloaded constructs like pop, push, and size that made using them a breeze. I thought I would run into some problems, I usually run into at least one road bump that causes my programs to not compile, but I rarely saw the program not compile. I was also very happy that my design from the previous assignments payed off and there were a number of core functions that I never had to touch. The combat system, used of weapons, and use of Plagued Creatures worked very well. I had to adjust the Plagued Creatures function slightly, but the rest of the functions worked great without me having to adjust them. Most of my work that I had to do was creating new functions, and reworking main to fit the tournament style. Overall, the programming was one of the smoothest of the term, which was encouraging after I struggled so much with lab 9. I would say the biggest challenge I had was getting the program to behave and display what I wanted it to, and finding out what the segmentation fault was in my testing phase.

I ended up having to make a few more functions to make the program more readable and to add some functionality. I added a function called addWins that was used to increase the number of wins a Creature has once it wins. This made the printStack function easier to work with because I could then compare the wins of each Creature at the top three spots of the stack. The printStack function became one of my longest functions and trickiest because I had to accurately determine which Creature got first, second, and third place. I initially thought it would be as easy as printing out the top three spots on the stack, however, if Creatures tie, there needs to be some kind of functionality that deals with that. I created a few if statements that checks to see if the top three have the same wins and then prints out the correct places afterward.

Another change I had to make that I hadn't thought of during the design phase was how to deal with the names of the Creatures. Each team would have the same types of Creatures so it could be confusing which Creature was attack and which team it was on. I change the names to Team (1 or 2), Creature (number of Creature on the team), (Name of Creature). This made it so you could identify what team the Creature was on, what number of Creature it is on that team, and what type of Creature it is. I edited this in the createCreature function, and I also made the same type of changes in the plaguedCreature function, so that if a Creature becomes plagued, it will have a similar name.

I also decided to only allow the teams to choose between 3 and 12 Creatures, beyond 12 Creatures the program would take a very long time to complete all of the fighting, and using less than 3 Creature per team doesn't seem like much of a tournament. I wanted to ensure that each tournament had a first, second, and third Creature, so there should be at least 4 Creatures fighting, but 6 total sounds better.

I think I should have put a little more thought into the standing system. I tried to research standing systems like the bcs, but this seemed a bit too complicated to put in a point system that determined which Creature came in what place. I do think, however, the the most important part to the user is whether or not their team wins, not which Creature did the best. I think if the users were able to have a library of Creatures that could level up and be used again and again if they stay alive would make the placing of the Creatures matter more, but in this simple of a program, it doesn't seem to make a big difference. This is one area I can improve on because there were a few problems with how the Creatures were assigned their places. It works really well most of the time because the game actually seems pretty balanced if you choose a bunch of different Creatures, but when one team destroys the other team and has more than three Creatures still alive, there is a possibility that some of the Creatures will not get the proper place. At least, there should be more ties with more Creatures included instead of just focusing on only the last three Creatures on the stack.

I wish I would have spent a bit more time looking at memory leaks as well. I'm not quite sure where I would have missed something that causes a memory leak, but I think if I would have focused a bit more on this before I started programming, I could have known how to stop the memory leaks. The program works and doesn't show many signs of the program leaking memory, but there was the one mysterious segmentation fault I received that I'm not sure how to fix. Any suggestions on this would be welcome.

Other than my previous comments, I think that the program I have designed and implemented, satisfies all of the requirements of the assignment, and I think shows my growth as a programmer after taking this class. I can't wait to see where this will all go, and I thank you for all your time and efforts in grading my material.