Makenzie Brian

November 20, 2015

CS 162

Assignment 4 Reflection

Problem Description:

Create a queue for each of the two teams. Create a stack for the losers. All creatures must eventually end up in the stack (the one who lost first will be in the bottom). The queue is FIFO. The stack is a FIFO. Runs a tournament with winner based on a last man standing concept (because the prompt is open ended) so the point system is really just the strength points of the characters. For each round which characters fought and who won will be displayed as well as the strength points because they are functioning as the point system in this scenario. At the end display the top 3 winners (the three who lasted the longest) and give an option to display the stack of losers. The rounds will be run using assignment 3 material. After a player wins a round it will go to the end of its teams stack and its health will increase by 50% of the damage that was inflicted.

Design Description:

There will be a struct called node with a pointer to next, an int for the team number, and a pointer to the information for the creature. The queue will need add, add initial, delete, empty checking, and display functions. It will also need a function to make the creature class accessible to the queue through the use of pointers (called first). It will need pointers to the nodes that are front and rear and an int for size. The stack will need add, empty checking, and display functions. It will also need a pointer to front and an int for size. The contents of the various functions is based on lab 6. Displays top 3 winners if there were more than 3 players, otherwise displays top 2. Also will put in a display for the tie (will not make a tie breaker because an arbitrary tie breaker defeats the purpose of having a tournament).

Reflection:

At the end of the rounds the last player is displayed with an adjusted strength point value from where a creature is normally put back into the queue with an adjusted strength from recovery time. I didn't fix this because they had already won so it wasn't relevant or important and doesn't affect who wins in the program. Otherwise the implementation is based on the design.

Test Plan:

Not including nicknames because they just help keep track of the characters. Also not including the option to see the losers because it operated as expected and doesn't affect the output of winners. I did not include all cases. Most of them operated as expected so I did not include all the testing I used, only base cases and exceptions to expected operation.

Input	Expected Output	Actual Output	Reason

Team1: Barb(a), Barb(b) Team2: Barb(c), Barb(d)	A team wins	Team2 wins, winners: d, a, b	normal
Team1: gollum(a), reptile(b), blue(c) Team2: Hydra(d)	A team wins	Team2 wins, winners: d, c, b	normal
Team1: reptile(a) Team2:Blue(b)	A team wins	Team1 wins, Winners: a,b	normal
Team1: gollum(a) Team2:Hydra(b), reptile(c)	Team2 wins	Team2 wins, winners: b,c	normal
Team1:blue(a), reptile(b) Team2: hydra(c)	A team wins	Team1 wins, winners: a,b	normal
Team1: Hydra(a) Hydra(b) Team2: Hydra(c), Hydra(d)	A team wins	Team2 wins, winners: c, b, d	normal
Team1:-1 Team2:-1	Seg fault	"Queue is empty. Don't be a butt." Seg fault(core dump)	Can't have a negative number of characters
Team1: a	Infinite loop	Infinite loop	No error checking because assuming user isn't stupid
Team1: 13	error	Error about pathways	Not a menu option so the program cannot process the input to an appropriate output
A team with 0 creatures	"Queue is empty. Don't be a butt." Seg fault	"Queue is empty. Don't be a butt." Seg fault(core dump)	It cannot run with teams of 0 creatures. Assuming the user isn't stupid.