CS 162 – Assignment 4
Fantasy Creature Tournament

Daniel Beyer
Dan.beyer@gmail.com
05/21/2016

Requirements

This program runs a creature tournament using the creature specifications from Assignment 3. A user will enter the number of fighters both players will use, then player supplies their lineup of creatures in order. After each round, the program needs to display which type of creatures fought and which won. At the end of the tournament the program will display: the first, second, and third place finishers, their teams, as well as which side scored more points. The program also needs to provide an option for the user to see a list of the final order of all fighters.
The program must use queues and stacks to hold the lineups and loser piles. When a creatures wins its round, it must have some percentage of its health restored with a heal function.

Type	Attack	Defense	Armor	Strength
Medusa	2d6*Glare	1d6	3	8
Barbarian	2d6	2d6	0	12
Baba Yaga	2d6*Soul	1d10	3	12
Blue Men	2d10	3d6*	3	12*Mob
Harry Potter	2d6	2d6	0	10/20*Hogwarts

*Glare: If Medusa rolls a 12 in attack, then the target dies.
*Soul: When Baba Yaga attacks, she receives 1/3 of her attack value (rounded down) towards her strength points.
*Mob: For every 4 points of damage (rounded down), they lose one defense die.
*Hogwarts: If Harry dies, he immediately recovers and his strength becomes 10. If he dies again, he dies for good.

Program Input

User enters:

- Initial number of creatures to participate in tournament
- Menu for Player 1 and Player 2 to select the creatures for their lineup:
    - 1. Barbarian
    - 2. Medusa
    - 3. Baba Yaga
    - 4. Blue Men
    - 5. Harry Potter
- - After each round, prompt user if they would like to see updated score and queues
- After the tournament is complete, option to press Y to view rest of lineups

Program Output

The program displays to the screen the progress of the fight between the two creatures. For each round of attack and defend, the program outputs the attack value, defense value, and the amount of damage done to the defending creature. Any special abilities that are performed are also displayed.

At the end of each round, the creature that died and the creature that are won are displayed, as is the amount of strength the winning creature healed. To help keep track of who is winning, the user is prompted to view each player's queue and the loser stack. At the end of the tournament, the top 3 finishers are displayed (based first on number of wins, then if those are tied, then based on remaining strength). The user is given the option to view the rest of the creatures from both teams.

Design

Main method pseudocode

```
Initialize random number seed, set to time(0) and seed srand
Declare int number of creatures

Prompt user for number of creatures
    - If invalid selection (>5), reprompt

Create game object
Use Game setPlayer functions to generate creature queues
Call Game combat function
Call Game results function
```

Creature class methods

Protected Variables:

- Int armor, strength, attackRolls, defRolls, attackSides, defSides, team, wins
- String name

Creature()
Default Constructor. Sets all int variables to 0 and the string variable to " ".

Creature(int armor, int strenth, string name, int attackRolls, int attackSides, int defRolls, int defSides)
Constructor for base class Creature. Sets variables to parameters specified.

string getName()
Getter function for name.

int getStrength()
Getter function for strength.

int getArmor()
Getter function for armor.

void setStrength(int)
Mutator function to set strength.

void setTeam(int)
Mutator function to set team.

void setWins()
Mutator function to set wins by incrementing 1.

int getWins()
Getter function for wins.

int getTeam()
Getter function for team.

void heal()

```
declare int heal
set heal equal to random number between 0 and 5
set strength equal to strength + heal
```

virtual int Attack() = 0
Pure virtual attack function (abstract class)

virtual int Defend(int attackValue) = 0
Pure virtual defend function (abstract class)

Barbarian class methods

Barbarian()
Constructor (use base class constructor)

int Attack()

```
initialize and set int attackValue to 0
for(int i =0; i<attackRolls; i++)
    - Add result of random roll to attackValue
Return attackValue
```

int Defend(int attackValue)

```
initialize and set int damage and defenseValue to 0
```

```
for(int i =0; i<defRolls; i++)
        -   Add result of random roll to defenseValue
If (attackValue < defenseValue + armor)
        -   Display “No damage done”
        -   Set damage = 0
Else
        -   Damage = attackValue – defenseValue – armor
If (damage >= strength)
        -   Strength = 0
Else
        -   Strength = strength – damage
Return damage
```

Medusa class methods

Medusa()

Constructor (use base class constructor)

int Attack()

```
initialize and set int attackValue to 0
for(int i =0; i<attackRolls; i++)
        -   Add result of random roll to attackValue
If(attackValue == 12)
        -   Medusa has used glare
        -   attackValue = 100 (to act as indicator for Main that glare was used)
Return attackValue
```

int Defend(int attackValue)

```
initialize and set int damage and defenseValue to 0
for(int i =0; i<defRolls; i++)
        -   Add result of random roll to defenseValue
If (attackValue < defenseValue + armor)
        -   Display “No damage done”
        -   Set damage = 0
Else
        -   Damage = attackValue – defenseValue – armor
If (damage >= strength)
        -   Strength = 0
Else
        -   Strength = strength – damage
Return damage
```

BabaYaga class methods

BabaYaga()

```
Constructor (use base class constructor)

int Attack()
initialize and set int attackValue to 0
for(int i =0; i<attackRolls; i++)
    -   Add result of random roll to attackValue
Set strength = attackValue/3 for Soul ability
Display “Baba Yaga used Soul ability to add attackValue/3 to her strength”
Return attackValue

int Defend(int attackValue)
initialize and set int damage and defenseValue to 0
for(int i =0; i<defRolls; i++)
    -   Add result of random roll to defenseValue
If (attackValue < defenseValue + armor)
    -   Display “No damage done”
    -   Set damage = 0
Else
    -   Damage = attackValue – defenseValue – armor
If (damage >= strength)
    -   Strength = 0
Else
    -   Strength = strength – damage
Return damage
```

BlueMen class methods

```
BlueMen()
Constructor (use base class constructor)

int Attack()
initialize and set int attackValue to 0
for(int i =0; i<attackRolls; i++)
    -   Add result of random roll to attackValue
Set strength = attackValue/3 for Soul ability
Display “Baba Yaga used Soul ability to add attackValue/3 to her strength”
Return attackValue

Int Defend(int attackValue)
initialize and set int damage and defenseValue to 0
for(int i =0; i<defRolls; i++)
    -   Add result of random roll to defenseValue
If (attackValue < defenseValue + armor)
    -   Display “No damage done”
```

```
        -    Set damage = 0
Else
        -    Damage = attackValue – defenseValue – armor
If (damage >= strength)
        -    Strength = 0
Else
        -    Strength = strength – damage
If (strength is greater than 4 and less than or equal to 8)
        -    defRolls = 2
        -    Display "BlueMen have 2 dice rolls due to Mob ability"
If (strength is greater than 0 and less than or equal to 4)
        -    defRolls = 1
        -    Display "BlueMen have 1 dice roll due to Mob ability"
Return damage
```

<u>HarryPotter class methods</u>

```
Public Variable:
        -    Bool hasDied
```

<u>HarryPotter()</u>

```
Constructor (use base class constructor)
        -    Set hasDied to false
```

<u>int Attack()</u>

```
initialize and set int attackValue to 0
for(int i =0; i<attackRolls; i++)
        -    Add result of random roll to attackValue
Set strength = attackValue/3 for Soul ability
Display "Baba Yaga used Soul ability to add attackValue/3 to her strength"
Return attackValue
```

<u>Int Defend(int attackValue)</u>

```
initialize and set int damage and defenseValue to 0
for(int i =0; i<defRolls; i++)
        -    Add result of random roll to defenseValue
If (attackValue < defenseValue + armor)
        -    Display "No damage done"
        -    Set damage = 0
Else
        -    Damage = attackValue – defenseValue – armor
If (damage is greater than strength and hasDied is false)
        -    Set strength to 10
        -    Display "Harry Potter used the Hogwarts ability"
        -    Set hasDied to true
If (damage is greater than strength and hasDied is true)
```

```
    - Strength = 0
Else
    - Strength = strength – damage
Return damage
```

HelperFunctions class methods

Menu()
Menu function to select creature. Displays list of creatures

Sleep(int milliseconds)
Pauses the program for a specified number of milliseconds

Queue class methods

Protected elements:

```
Struct Queuenode
    - Declare pointer to previous and next Queuenodes
    - Declare pointer to creature
    - Queuenode(Creature* cr, Queuenode* next, Queuenode* previous)
        o Constructor for Queuenode struct
        o If no parameters for next and previous, set to NULL
        o Otherwise set to declared pointers next and previous
    - Declare pointer to front and back Queuenodes
    - Declare int size
```

Queue()
Default Queue constructor, sets front and back pointers to NULL

~Queue()
Destructor for Queue

```
Set Queuenode pointer temp = front pointer
While (temp is not NULL)
    - Set Queuenode pointer garbage = temp
    - Set temp to temp->next
    - Delete the creature in garbage
    - Delete garbage node
```

Void add(Creature*)

```
If(front pointer is NULL)
    - Front = new creature Queuenode
    - Back equals front
Else
Queuenode pointer = new creature Queuenode
Pointer-> next = NULL
Back->next = pointer
Back equals pointer
```

Creature* remove()

```
if(front equals NULL)
    - Queue is empty
Else
Set temp creature pointer to NULL
Set temp Queuenode pointer to front
Set creature pointer to temp Queuenode pointer value
Set front to point to next node
If(front does not equal NULL)
    - Set previous node to NULL
Delete temp node
Return creature pointer
```

Void display()

For testing purposes and grader accessibility.
Display contents of Queue

Bool isEmpty()

```
if(front is NULL)
    - Return true
Else
Return false
```

Int getSize()

Getter function for size.

Void setSize()

Mutator function to set size.

Void decSize()

Mutator function that decrements size by 1.

Void sort()

```
Declare and initialize Queuenode* current = front, prev = NULL, tempNode = NULL
Declare and initialize int changeFlag = 0
For (int i = 0, i<size of Queue, i++)
    - While(next node is not NULL)
        o tempNode = current->next
        o if(current node creature value of getwins is > tempNode's creature getwins)
            ▪ changeFlag = 1
            ▪ set current->next equal to tempNode->next
            ▪ set tempNode->next equal to current
        o  if(prev does not equal NULL)
               ▪  prev->next equals tempNode
        o  set prev = tempNode
        o  if(front is current)
               ▪  set front equal to tempNode
        o  if(current->next is NULL)
               ▪  set back equal to current
        o  else
               ▪  set prev equal to current
               ▪  set current equal to current->next
        o  if(changeFlag is 0)
               ▪  break out of function
        o  else
               ▪  set prev equal to NULL
               ▪  set current equals front
               ▪  set changeFlag to 0
```

<u>Stack class methods</u>

<u>Protected elements:</u>

```
Struct Stacknode
    -  Creature* value
    -  Stacknode* next
    -  Stacknode(Creature* val, Stacknode* nxt = NULL)
        o  Set value equal to val
        o  Set next = nxt
Stacknode* top
```

<u>Stack()</u>

Default constructor, sets top to NULL

<u>~Stack()</u>

```
Destructor for Stack
Set Stacknode pointer temp = front pointer
While (temp is not NULL)
    -  Set Stacknode pointer garbage = temp
    -  Set temp to temp->next
    -  Delete the creature in garbage
    -  Delete garbage node
```

<u>Void add(Creature*)</u>

```
if(top is NULL)
    -  Top equals new Stacknode(creature)
Else
```

```
    - Top equals new Stacknode(creature, top)
```

<u>Creature* remove()</u>

```
Declare Stacknode* temp
Declare and initialize Creature* tempCr equal to NULL

If(top is NULL)
    - Stack is empty
Else
Temp equals top
tempCr equals top stack value
delete temp
return tempCr
```

<u>Void display()</u>

For testing purposes and grader accessibility.
Display contents of Queue

<u>Bool isEmpty()</u>

```
if(top is NULL)
    - Return true
Else
Return false
```

<u>Game class methods</u>

<u>Protected elements:</u>

Creature* creature, creature1, creature2
Queue player1, player2
Stack losers
Int player1pts, player2pts

<u>Game()</u>

Default construct, sets creature, creature1, and creature2 to NULL
Player1pts and Player2pts = 0

<u>Void setPlayer1(int)</u>

```
Declare and initialize int i = 0
Set player1 size
Prompt user to select creatures for their team
While(i<number of creatures)
    - Declare int choice and string name
    - Display menu()
    - While choice invalid
        o Reprompt
    - Prompt user to enter a custom name
    - If (choice ==1)
        o Create new Barbarian
    - If (choice == 2)
        o Create new Medusa
    - If (choice == 3)
        o Create new BabaYaga
    - If(choice == 4)
        o Create new BlueMen
    - If (choice == 5)
        o Create new Harry Potter
    - Set creature team
    - Add creature to player1 queue
```

<u>Void setPlayer2(int)</u>

```
Declare and initialize int i = 0
Set player2 size
Prompt user to select creatures for their team
While(i<number of creatures)
    - Declare int choice and string name
    - Display menu()
    - While choice invalid
        o Reprompt
    - Prompt user to enter a custom name
    - If (choice ==1)
        o Create new Barbarian
    - If (choice == 2)
        o Create new Medusa
    - If (choice == 3)
        o Create new BabaYaga
    - If(choice == 4)
        o Create new BlueMen
    - If (choice == 5)
        o Create new Harry Potter
    - Set creature team
    - Add creature to player2 queue
```

<u>Void Combat()</u>

```
Set int round = 0
While (neither player queue is empty)
    - Creature1 = player1.remove()
    - Creature2 = player2.remove()
    - Set int starting strength variables for each creature to their initial strength
    - Display Customname "fights" customname for each creature
- While (both creatures' strength > 0)
    o Display creature2 remaining strength
    o Creature1 attacks creature2, returns attackValue
        ▪ If attackValue == 100, Medusa used Glare
            • Creature2->setStrength(0)
            • Creature2 died
        ▪ Else
            • Display attackValue
            • Creature2 defends against creature1, returns damage
            • Display defense roll and damage taken
        ▪ If creature2-<getStrength()<=0
            • Creature 2 has died
            • Creature 1 wins
            • Creature1->setWins
            • Add creature2 to losers stack
            • Decrease player2 queue size
            • Heal creature1
            • If(creature1 strength > starting strength)
                o Set creature1 strength to starting strength
            • Add creature1 back to queue
            • Incremement player1pts
    o Creature2 attacks creature1, returns attackValue
        ▪ If attackValue == 100, Medusa used Glare
            • Creature1->setStrength(0)
            • Creature1 died
        ▪ Else
            • Display attackValue
            • Creature1 defends against creature2, returns damage
            • Display defense roll and damage taken
        ▪ If creature1-<getStrength()<=0
            • Creature 1 has died
            • Creature 2 wins
            • Creature2->setWins
            • Add creature1 to losers stack
            • Decrease player1 queue size
            • Heal creature2
            • If(creature2 strength > starting strength)
                o Set creature2 strength to starting strength
            • Add creature2 back to queue
            • Incremement player2pts
- Increment round
```

```
Void results()
If(player1pts > player2pts)
        -  Sort player1 queue
        -  Add creatures from player1 queue to losers stack
If(player2pts > player1pts)
        -  Sort player2 queue
        -  Add creatures from player2 queue to losers stack
Set Creature* winner, second, and third pointers to losers.remove() respectively
If (winner->getWins() == second->getWins() == third->getWins()) or (winner->getWins() == second->getWins()) or (second->getWins() == third->getWins())
        -  Sort based on remaining strength
Display First Place:
Display Second Place:
Display Third Place:

Prompt user if they would like to view remaining creatures
If Yes:
While (losers.isEmpty is false)
Creature* creature = losers.remove()
Display all remaining creatures
```

Test Plan

Test Case	Input Values	Driver Functions	Expected Outcome	Observed Outcome

Input out or range or not an integer	Input < 1 or input "a"	Main()
		while (numCr<1 || !cin)
			"Invalid selection, please select a number greater than 1"	"Invalid selection, please select a number greater than 1"

Barbarian vs Barbarian	Input = 1
	Input = 1
		Main()
		Attack()
		Defend()
			Barbarian does random attack damage 2-12 and random defense of 2-12.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				Barbarian does random attack damage 2-12 and random defense of 2-12.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Barbarian vs Medusa	Input = 1
	Input = 2
		Main()
		Attack()
		Defend()
			Barbarian does random attack damage 2-12 and random defense of 2-12.
			Medusa does random attack
				Barbarian does random attack damage 2-12 and random defense of 2-12.
				Medusa does random attack

			damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.	damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken. **Glare observed

Barbarian vs Baba Yaga	Input = 1
	Input = 3
		Main()
		Attack()
		Defend()
			Barbarian does random attack damage 2-12 and random defense of 2-12.
			Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				Barbarian does random attack damage 2-12 and random defense of 2-12.
				Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Barbarian vs Blue Men	Input = 1
	Intput = 4
		Main()
		Attack()
		Defend()
			Barbarian does random attack damage 2-12 and random defense of 2-12.
			Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12 while (4<strength<=8), and random defense 1-6 while strength <=4.
				Barbarian does random attack damage 2-12 and random defense of 2-12.
				Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12 while (4<strength<=8), and random defense 1-6 while strength <=4.

			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.	Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Barbarian vs Harry Potter	Input = 1
	Intput = 5
		Main()
		Attack()
		Defend()
			Barbarian does random attack damage 2-12 and random defense of 2-12.
			Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				Barbarian does random attack damage 2-12 and random defense of 2-12.
				Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Medusa vs Medusa	Input = 2
	Intput = 2
		Main()
		Attack()
		Defend()
			Medusa does random attack damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken
				Medusa does random attack damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken
				**Glare observed

Medusa vs Baba Yaga	Input = 2
	Intput = 3
		Main()
		Attack()
		Defend()
			Medusa does random attack damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies.
			Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				Medusa does random attack damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies.
				Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				**Glare observed

Medusa vs Blue Men	Input = 2
	Input = 4
		Main()
		Attack()
		Defend()
			Medusa does random attack damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies.
			Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12 while (4<strength<=8), and random defense 1-6 while strength <=4.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				Medusa does random attack damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies.
				Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12 while (4<strength<=8), and random defense 1-6 while strength <=4.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

				**Glare observed

Medusa vs Harry Potter	Input = 2
	Input = 5
		Main()
		Attack()
		Defend()
			Medusa does random attack damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies.
			Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				Medusa does random attack damage 2-12 and random defense of 1-6. If she rolls 12 attack, Glare performed, attackValue = 100, opponent dies.
				Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				**Glare observed

Baba Yaga vs Baba Yaga	Input = 3
	Input = 3
		Main()
		Attack()
		Defend()
			Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Baba Yaga vs Blue Men	Input = 3
	Input = 4
		Main()
		Attack()
		Defend()
			Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.	Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.

			Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12 while (4<strength<=8), and random defense 1-6 while strength <=4. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.	Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12 while (4<strength<=8), and random defense 1-6 while strength <=4. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Baba Yaga vs Harry Potter	Input = 3
	Input = 5
		Main()
		Attack()
		Defend()
			Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.
			Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead.
			Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.
				Baba Yaga does random damage 2-12 and random defense 1-10. For every attack, she adds 1/3 of attack value to her strength.
				Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead.
				Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Blue Men vs Blue Men	Input = 4
	Input = 4
		Main()
		Attack()
		Defend()
			Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12	Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12

			while (4<strength<=8), and random defense 1-6 while strength <=4. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.	while (4<strength<=8), and random defense 1-6 while strength <=4. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Blue Men vs Harry Potter	Input = 4
	Input = 5
		Main()
		Attack()
		Defend()
			Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12 while (4<strength<=8), and random defense 1-6 while strength <=4. Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.	Blue Men does random attack damage 2-10 and random defense 3-18 while strength > 8, random defense 2-12 while (4<strength<=8), and random defense 1-6 while strength <=4. Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Harry Potter vs Harry Potter	Input = 5
	Input = 5
		Main()
		Attack()
		Defend()
			Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.	Harry Potter does random attack damage 2-12 and random defense of 2-12. When he dies the first time, his strength = 10 and he continues. When he dies the second time, he stays dead. Damage taken is attackValue – defenseValue – armor. If attackValue is less than defenseValue+armor, no damage is taken.

Heal() function	None	Heal()	Each winning creature heals a random amount between 0 and their remaining health at the end of the round. If they heal more than their initial starting health, their new health is the same as their initial starting health.	Each winning creature heals a random amount between 0 and their remaining health at the end of the round. If they heal more than their initial starting health, their new health is the same as their initial starting health.

Queue add function	Add 4 creatures with unique names	Queue:
		Add()
		Display()
		Remove()
			For each creature chosen from the menu, user inputs custom name, display() function outputs the contents of the queue showing the list of creatures.	For each creature chosen from the menu, user inputs custom name, display() function outputs the contents of the queue showing the list of creatures.

Stack add function	None	Stack:
		Add()
		Remove()
			For each creature that loses, it is added to Stack. Display() function displays stack contents showing losing creatures	For each creature that loses, it is added to Stack. Display() function displays stack contents showing losing creatures

Queue sort function	None	Queue: Sort()	Queue is rearranged so the creature with the most wins is at the front of the	Queue is rearranged so the creature with the most wins is at the front of the

			Queue. Ties are not sorted by any other characteristic	Queue. Ties are not sorted by any other characteristic
Stack remove function	None	Stack remove()	Creature is removed and returned from the top of the stack	Creature is removed and returned from the top of the stack
Game results function	None	Game results()	Winning player is displayed. The top 3 finishers are sorted, first by wins. If there are ties, they are sorted by remaining strength. User is prompted to view remaining creatures. If YES, the remaining creatures from both teams are displayed from the stack	Winning player is displayed. The top 3 finishers are sorted, first by wins. If there are ties, they are sorted by remaining strength. User is prompted to view remaining creatures. If YES, the remaining creatures from both teams are displayed from the stack

2 Baba Yaga tie	None	Attack()
		Defend()
		Combat()
			If 2 Baba Yaga creatures fight, their SOUL ability can allow infinite strength gain. To prevent this, the first creature to reach 30 strength wins.	2 Baba Yaga creatures fight. First creature to reach 30 strength wins.

Reflection (And more notes on Testing Results)

My initial design for adding creatures to the player queues and running the combat worked well. Where I ran into the most difficulty was sorting the winning player queue after the tournament. I toyed with the idea of creating a vector to store sorted creatures, but I remember the professor dissuading us from using vectors for this part of the course. I decided on my final implementation of sorting the queue while still in the queue and then transferring the newly sorted queue to the stack to be displayed. This seemed to work well. Once the queue is passed to the stack, ties in wins between the top 3 finishers are actively sorted by strength remaining, and then they are finally displayed in order of First Place, Second Place, and Third Place.

To determine the sorting is working correctly, I display the wins and remaining strength for each creature.

To test this, I ran multiple tournaments until I had results of top 3 finishers with the same number of wins. I verified that for these top 3 finishers with tied number of wins, they were sorted by remaining strength.

Example Result:

- FIRST PLACE: Barbarian barb1, Team 1, Wins: 2, Remaining strength 12
- SECOND PLACE: Blue Men blue1, Team 1, Wins: 1, Remaining strength 12
- THIRD PLACE: Medusa med1, Team 1, Wins: 1, Remaining strength 8

To further verify that the creatures from the winning queue were being correctly sorted and then placed on the stack for viewing after the tournament, the remaining creatures are displayed with their number of wins and remaining strength (among other attributes).

I also ran into a problem with creatures regaining more health than they initially started with at the end of a winning round. This caused a problem with creatures becoming too powerful and then never losing. After noting the requirement that all creatures must not regain more health than they started with, I implemented the current solution of setting an int variable startingStrength at the beginning of each round to equal the starting strength of each creature. If the creature's health goes over this value after healing at the end of the round, their strength is returned to that starting strength value. On a similar problem, I found that if two Baba Yaga creatures fought they would get into an infinite loop of increasing strength due to their SOUL ability. To resolve this, I allowed the first creature to reach 30 strength to be the winner.

A big issue I ran into at the end of this project was memory leaks. Running my program with valgrind revealed problems with how I was cleaning up my pointers and objects. I eventually resolved the problem by iterating through the stack at the end of the Game results() function and manually removing its contents. Running the program with valgrind –leak-check=yes revealed no memory leaks after including this solution.

This assignment gave me a great opportunity to practice more with using pointers and to become much more familiar with queues, stacks, and using structs. It also allowed me to see how powerful and elegant polymorphism can be when creating a program many different classes.