Nicole Parker
Assignment 3

<u>Design:</u>

For this program it was obvious that I needed a base class, I decided to call this class Creature. In addition, to the base class I also needed five subclasses for each type of creature types: Golbin, Shadow, Barbarian, BlueMen, and ReptilePeople. It made the most sense that the base class would contain the variables for type, strength, and armor. It would also contain functions to attack, defend, applydamage and check if an opponent is dead, because these would be inherited to all subclasses. After thinking about the special features, specifically the “Achilles” further, I decided in order to account for this condition I would need a variable that would be applied to an opponent if the Goblin rolled a 12 for attack. I decided to make it a part of the base class so that it would affect the results for all additional attacks that occurred after it was updated. In addition, it would also make it so if the Goblin rolled a 12 for attack again, an additional multiplier would be added.

I decided that the best way to account for the Shadow's special defense feature was to create a two sided die, and if the result was even then it would take no damage. This would account for the 50/50 nature of shadow being in a different location than where an opponents attack was directed. The two sided die would essentially act as a coin flip.

I also decided that re-using the code for lab4 would be really helpful. It essentially had all of the components I needed. I was able to add in a function to roll three dice and return the sum. Which was necessary for reptile and blue men. I also decided that making rollAttackDice(), rollDefenseDice() pure virtual functions, allowing for each subclass to implement its unique dice requirements, and then just pass this result back into a variable, that would be used in the attack() function. This seemed more efficient than changing the entire attack function for each subclass. The basic outline I came up with is below.

-This is the base class. It has a type, strength, armor and multiplier(Goblin's special ability). rollAttackDice() and rollDefenseDice() will be pure virtual so each subclass will have to override these with their special conditions.

```
1. class Creature
protected:
string type
int strength
int armor
float multiplier
virtual int rollAttackDice() = 0
virtual int rollDefenseDice() = 0

public:
virtual void attack(Creature &opponent)
virtual int defense()
virtual void applyDamage(int damage)
bool isDead()
```

-For the Dice class it will pretty much be everything from lab4. I will need to add in a function to rollThreeDice(), I also don't need the loadedDice function. This will be used to get the attack, and defense amounts, which will ultimately result in total damage

2.
class Dice
protected:
int numSides

public:
Dice()
Dice(int numSides)
int rollDice()
int rollTwoDice(Dice& die1, Dice& die2)
int rollThreeDice(Dice& die1, Dice& die2, Dice& die3)

-Each subclass will have a rollAttackDice() and rollDefenseDice() function which will follow their specific attack and defense guidelines. For the shadow it will have to contain a virtual applyDamage(), which will have a two sided dice and if an even number is rolled then the shadow will not be damaged. Goblin will have a virtual attack() function that will account for the special feature "Achilles". In this case if a we is rolled and goblin is not fighting a goblin then a multiplier will cut the opponent's strength in half, and remain at half for the rest of the battle.

<u>Testing:</u>
The key things I wanted to test were that the applyDamage() function calculated and applied damage correctly, that the creature did die when appropriate, and that the two special cases of "achilles" and the shadow defense worked.

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
Verify that attack values, armor, defense and damage all fall within the appropriate ranges	None	None	The attack values, armor, defense and damage all fall within the appropriate ranges	The attack values, armor, defense and damage all fall within the appropriate ranges
The applydamage function subtracts the correct amount of damage from the correct player	None	None	The new strength for the player reflected the correct amount to be subtracted based on players defense and armor	The new strength for the player reflected the correct amount to be subtracted based on players defense and armor
- The "Achilles" attack for Goblin applied multiplier when a 12 was rolled	None	None	When a 12 is rolled by Goblin, the opponent's attacks would be cut in half	- The damage multiplier wasn't initially calculated properly. I discovered that it was because of

				integer division problems. I changed the multiplier to a double, and this resolved the issue.
Test the Shadow special defense is implemented 50% of the time	None	None	For 50% of rolls the shadow will no take any damage because he will no be in the place of attack.	For 50% of rolls the shadow will no take any damage because he will no be in the place of attack.

<u>Reflection:</u>

There were a few things that I overlooked in my design. One of these was how the Dice class would work in relation to the creature class. At first I tried to instantiate the necessary dice objects for each subclass. This did not make much sense after looking it over. I found that a better solution was to make the functions for rollTwoDice, and rollThreedice static. This meant that because the static function was not attached to a particular object, it could be called directly.

I also found that because I was deleting instances of a class through a pointer to the parent the complier was having issues with there being no destructor. I had to create a virtual destructor in order to compile.

In writing the code I also realized that it would make more sense to make attack take a pointer to an Creature opponent. By doing this attack would be more versatile, and would be able to evaluate some very helpful aspects of the opponent. An example of this is in relation to the Goblin class, having the opponent pointer allows it to easily check if the goblin is fighting a type goblin which is necessary to check if the special feature of “Achilles” can be applied. In addition, this also makes it so the code for Goblin's special attack does not need to be in another function, everything can be done in a virtual attack function.

The last big piece that I didn't really think through was how to design int main. I did know that I would need to have multiple scenarios that needed to take place such as Goblin vs Goblin, Goblin vs Shadow etc. but I didn't think it though much more than this. After writing the code to run one scenarios I realized that repeating all of this code again for each match up would be redundant. I found that making a function which I called fight, which took parameters for a player 1 and player2 and then executed the necessary attack functions would be much cleaner. I would then be able to call this function for each scenario saving a lot of time and making the program easier to read.

With regards to the mechanics of the game I realized that player1 is given a significant advantage by going first. So to compensate for this I allowed player2 to roll an attack even though they may be mortally wounded. This made possible for a double kill scenario.