Assignment 4 Design
Tatyana Vlaskin

1. First modification that I need to make is to make my parent class.
2. Two functions are used by 4 child classes: int attack() and int defense().
3. These functions are slightly different in each child class, but have the same number of parameters. I will redefine the function in each child class and make these functions virtual. This will allow the base point and use the most-derived version of the function that it finds.
4. So the parents class will look somewhat like this. Please note that I'll make attack() and defense() functions pure virtual, by setting them equal to zero:

```
  class CHARECTOR{
protected:
  string name;
  int atackNumberOfDiceRoles;-- will get rid of this variable, see below for explanation
  int atackDiceSides; ;-- will get rid of this variable, see below for explanation
  int armor;
  int defenceNumberOfDiceRoles; ;-- will get rid of this variable, see below for
explanation
  int defenceDiceSides; ;-- will get rid of this variable, see below for explanation
  int damage;
  int strength;
  int ststore;// this variable will be used to calculate the % of the strenght points left
public:
int getarmor(){return(armor);}
int getstrength(){return(strength);}
void setstrength(int strength2){    strength=strength2;    }
```
virtual int attack() =0;{
virtual int deffence()=0;{
```
int getstnstore(){return(ststore);}
int getdamage(){return(damage);}
string getname(){return(name);}
```

5. Next change will be to modify my child classes.
6. Currently my child classes have on constructor in them.
7. Each class will get 2 virtual functions. One for attack and one for defense.
8. My current attack function in the parent class look like this:

```
    int attack(){
     int sum=0;
     int randomAttackPoints;// variable to keep track of atack points
     cout << endl << name << " - attack ";// will display a message who is attacking
```

```
      for(int i=0; i<atackNumberOfDiceRoles; i++){// the dice will be rolled as many times as indicated for each character
          randomAttackPoints = rand()%atackDiceSides + 1; // the values of attack will be between 1 and sides of atack points
          cout << randomAttackPoints << " ";// message that will display how many atack points are generated
          sum+=randomAttackPoints;
      }
      cout<<"Sum : " << sum << endl;// message that will indicate total attack points
      return(sum);
    }
```

9. AtackDiceSides and **atackNumberOfDiceRoles variables will be removed from the base class and in the actual attack functions they will be replaced with the appropriate values in each class during the function redefinition.**
10. **Similarly** int defenceNumberOfDiceRoles and int defenceDiceSides variable will be deleted from the base class and in the defense functions they will be replaced with the values corresponding to each character.
11. As the result of the just described change, the constructor of the child classes, needs to be changed as well. AtackDiceSides, **atackNumberOfDiceRoles,** defenceNumberOfDiceRoles and int defenceDiceSides variable will be removed.
12. New child classes will look somewhat like that

```
///CLASS GOBLIN
class Goblin:public CHARECTOR{
public:
Goblin(){name="Goblin"; armor=3; strength=ststore=8; }
virtual int attack(){
for(int i=0; i< 2 ; i++){// need to roll attack dice x2
        randomAttackPoints = rand()%6 + 1; // 6 sides on the attack dice
}
virtual int deffence(){
for(int i=0; i< 1 ; i++){// need to roll deffene dice x1
        randomDeffencePoints = rand()%6 + 1; // 6 sides on the defence dice
}

///CLASS BARBARIAN
class Barbarian:public CHARECTOR{
public:
  Barbarian(){name="Barbarian";armor=0;strength=ststore=12}
virtual int attack(){
for(int i=0; i< 2 ; i++){// need to roll attack dice x2
```

```
        randomAttackPoints = rand()%6 + 1; // 6 sides on the attack dice
}
virtual int deffence(){
for(int i=0; i< 2 ; i++){// need to roll deffene dice x2
        randomDeffencePoints = rand()%6 + 1; // 6 sides on the defense dice
}
///CLASS REPTILEPEOPLE
class ReptilePeople:public CHARECTOR{
public:
  ReptilePeople(){name="ReptilePeople";armor=7;strength=ststore=18; }
virtual int attack(){
for(int i=0; i< 3 ; i++){// need to roll attack dice x3
        randomAttackPoints = rand()%6 + 1; // 6 sides on the attack dice
}
virtual int deffence(){
for(int i=0; i< 1 ; i++){// need to roll deffene dice x1
        randomDeffencePoints = rand()%6 + 1; // 6 sides on the defence dice
}
///CLASS BLUEMEN
class BlueMen:public CHARECTOR{
public:
  BlueMen(){name="BlueMen";armor=3;strength=ststore=12;}
virtual int attack(){
for(int i=0; i< 2 ; i++){// need to roll attack dice x2
        randomAttackPoints = rand()%10 + 1; // 10 sides on the attack dice
}
virtual int deffence(){
for(int i=0; i< 3 ; i++){// need to roll deffene dice x3
        randomDeffencePoints = rand()%6 + 1; // 6 sides on the defence dice
}
```

13. Next step will to be create one new character class. Ill create a Mega Man. He will be derived from the Barbarian class. Description of the Mega Men from the previous assignment is provided below.

    Mega Man

    Any damage that would normally be applied to strength gets diverted to an energy store. The energy store can be used to make an enhanced attack. Any damage received in excess of the store is applied to strength as usual.

14. After reading the description it become obvious that we need to introduce new variable, lets call it EnergyStore. Additionally, let's introduce one more attack function. This attack function will use enchanted weapon.

```
  ///CLASS MEGA MEN
class MegaMen:public Barbarian{
public:
//I'll randomly choose armor, strength and EnergyStore of the Mega men
//also attack function will be inherited from the barbarian and will not be redefined
  MegaMen (){name=" Mega Men ";armor=10;strength=ststore=20; EnergyStore =10;}
virtual int deffence(){
for(int i=0; i< 2 ; i++){// need to roll deffene dice x2
        randomDeffencePoints = rand()%6 + 1; // 6 sides on the defense dice
}
```

The main difference between this class and any other character classes will be that the impact on health after attack is different.

Any damage that would normally be applied to strength gets diverted to an energy store. The energy store can be used to make an enhanced attack. Any damage received in excess of the store is applied to strength as usual.

As a result of this when the strength of the MegaMen is calculated after the battle, it will be done somewhat like this:

```
int netAttackPoints=player.attack()-MegaMen.deffence();    netAttackPoints=
netAttackPoints - MegaMen.getarmor();

   if(netAttackPoints<0){
      netAttackPoints=0;
   }

MegaMen NewEnergy= MegaMen.getstrength()-netAttackPoints;

   if(MegaMen NewEnergy <0){
            MegaMenNewEnergy = absolute value of (MegaMenNewEnergy)                    Meg

  }
```

And finally, we need to reset strength and energy store:

```
MegaMen.setstrength(MegaMenNewStrength);

MegaMen.setstrength(MegaMenNewEnergy);
```

15. Because of the MegaMen, it becomes obvious that we need to create a way to use enchanted weapon. In order to user enchanted weapon, the character needs to have energy store points- see blue man for description. As a result of this, Ill add a variable to the base class and call it Energy. Each character will get some energy points that they will be able to use to fight with enchanted weapon. Once, the character runs out of the energy points, they will not be able to use enchanted weapon anymore and will be forced to use regular attack function. MegaMen is the only character that will be able to use energy for defense as well. All other characters will be able to use energy only during the attack. Each time an enchanted weapon is used, a character loses 1 energy point.
    a. **///CLASS GOBLIN - 3 energy points**
    b. **///CLASS BARBARIAN -4 energy points**
    c. **///CLASS REPTILEPEOPLE-5 energy points**
    d. **///CLASS BLUEMEN-6 energy points**
    e. **///CLASS MEGA MEN-10 energy points**
16. Ok back to the enchanted weapon. During the combat, a character will be asked if they want to use regular weapon or enchanted weapon. If they have energy points they will be allowed to use enchanted weapon that will impose x2 points of attack. Same attack function will be used. However, whatever value is produced by the attack function will be multiplied by 2 and that will be the resultant attack points. Attack points will be subtracted from the strength points. Please note only Mega Man will be able to use energy during defense.
17. Next, I need to introduce venom. According to the description that was provided in the assignment, we need to do something like this:

    A single bite, sting, spit whatever inserts venom in the target. For a specified nu mber of turns applies constant or degrading damage to the target's strength. As it's not physical energy
    Mega Man cannot divert the damage to the energy store but is taken against str ength.

    This does not make such sense to me. What I will do is that if the character is attacked with the venom, their defense and attack points will be reduced by certain %.

I think to make testing easier, Ill give each character only 1 bottle of venom.

The will be an option where each character is asked if they want to attack with venom. No damage to the health of the opponent will be done, but their defense and attack points will be reduced by certain % depending who imposes a venomous damage for the next 2 rounds. Thus, there will be another variable added into the classes, lets call it VenomeDamage.

a. **///CLASS GOBLIN - 5% imposed damage**
b. **///CLASS BARBARIAN -5% imposed damage**
c. **///CLASS REPTILEPEOPLE-20% imposed damage**
d. **///CLASS BLUEMEN-7% imposed damage**
e. **///Mega Men-15% imposed damage**

18. One more think that will be added to the program is spinach. To make testing simple, Ill let the character each the spinach before the combat. If the character decides to eat spinach, their attack points will quadruple, and maybe finally, my goblin will be able to win. It was very weak character in the last battle game. As a result of the spinach, each class will have another attack function that will reflect spinach.
19. Another change that I need to make to my program is make sure that base class is never called, thus, I need to change my switch statements:

I'll delete the following 2 objects: CHARECTOR player,enemy;

```
Goblin goblin1;
  Barbarian barbarian1;
  ReptilePeople reptile1;
  BlueMen blueMen1;
  MegaMen megaMen1;

  CRectangle rectangle;
  CTriangle triangle;

  CHARECTOR * ptr_ goblin  = & goblin1;
  CHARECTOR * ptr_ barbarian = & barbarian1;
  CHARECTOR * ptr_ reptile = & reptile1;
  CHARECTOR * ptr_ blueMen = & blueMen1;
  CHARECTOR * ptr_ megaMen = & megaMen1;

   switch(result){
     case 1:
           CHARECTOR * ptr_ goblin  = & goblin1;
```

```
            break;
          case 2:
            CHARECTOR * ptr_ barbarian = & barbarian1;
             break;
      etc.
```

As a result of this when I'll try to get attack or defense functions, I need to do something like this:

```
  ptr_ goblin  ->defence ()
```

Testing:

In summary:

The following battle combinations need to be tested with all commendations of characters. Lest use barbarian and goblin as an example.

1. None of the characters use spinach, none of the characters use enchanted weapons and none of the characters use venom
2. Goblin uses spinach, barbarian does not use spinach, neither goblin nor barbarian use enchanter weapon or venom
3. Barbarian uses spinach, goblin does not use spinach, neither goblin nor barbarian use enchanter weapon or venom
4. None of the characters use spinach, barbarian uses enchanted weapon until runs out of energy point, goblin uses only regular attack and none of the characters use venome
5. None of the characters use spinach, goblin user enchanted weapon until runs out of energy points, barbarian does not use enchanted weapon, none of the characres us venom
6. None of the characters use spinach, both characters use enchanted weapons until they run out of energy , neither goblin nor barbarian use enchanter weapon or venom
7. None of the characters use spinach, both characters use enchanted weapon randomly – this is complicated.
8. Do tests indicted in 4-7, when both characters use spinach
9. Do tests indicated in 4-7 when only barbarian uses venom
10. Do tests indicated in 4-7 when only goblin uses venom
11. Do tests indicated in 4-7 when both goblin and barbarian use venom
12. Do step 8 when only barbarian uses venome
13. Do step 8 when only goblin uses venom
14. Do step 8 when both goblin and barbarian use venom
15. Do the above mentioned tests for different character combinations.

I have to do similar testing that were done in the last battle game, so I am copying and pasting the chart from the last game here.

Whats are we testing	How we are testing	What is expected	What is the output	PASS/FAIL
Make sure that radom number of attack points is generated each time	cout << randomAttackPoints << " sum+=randomAttackPoints;	Each time there is a different attack points		
Make sure Goblin rolls the dice x2 for attack	There is a for loop with the : cout << randomAttackPoints << " sum+=randomAttackPoints;	2 values for attack should be displayed		
Make sure Goblin gets values from 1-6 for each dice roll	There is a for loop with the : cout << randomAttackPoints << " sum+=randomAttackPoints;	Values of attack1 and attack 2 between **1-6**		
Make sure Goblin attack1 and attach2 value add up and assigned to total attack points	There is a for loop with the : cout << randomAttackPoints << " sum+=randomAttackPoints;	Attack will be equal to sum of 2 dice roles		
Do similar tests described for Goblin	See Goblin	See goblin		

for Barbarian				
Do similar tests described for Goblin for Reptile	Similar to Goblin, the difference is that, there have to be 3 rolls of dice and dice values are from 1 to 3. There is a for loop with the : for(int i=0; i<atackNumberOfDiceRoles; i++){ cout << randomAttackPoints << " sum+=randomAttackPoints	Make sure that are attack1, attack2 and attack3 have value and they are in the range **<u>1-3</u>**		
Do similar tests described for Goblin for Blue Man	Similar to Goblin, the different is that, there have to be 2 rolls of dice and dice values are from 1-10.	Make sure that are value for attack 1 and attack2 and values are between **<u>1-10</u>**		
Make sure Goblin rolls the dice x1 for defense	for(int i=0; i<defenceNumberOfDiceRoles; randomDeffencePoints = rand()%defenceDiceSides+1; cout << randomDeffencePoints << " "; sum+=randomDeffencePoints; }	Defense have value		
Make sure Goblin gets values from 1-6 for each dice roll for defense	for(int i=0; i<defenceNumberOfDiceRoles; randomDeffencePoints = rand()%defenceDiceSides+1; cout << randomDeffencePoints << " "; sum+=randomDeffencePoints; }	Values of defense is between **<u>1-6</u>**		
Do similar tests described in Goblin for Barbarian. The	for(int i=0; i<defenceNumberOfDiceRoles; randomDeffencePoints = rand()%defenceDiceSides+1; cout << randomDeffencePoints << " "; sum+=randomDeffencePoints;	Values of defense is between **<u>1-6</u>**		

difference is that barbarian needs to roll dice x2 for defense	}			
Make sure barbarian defence1 and defense values add up and to total defense points	for(int i=0; i<defenceNumberOfDiceRoles; randomDeffencePoints = rand()%defenceDiceSides+1; cout << randomDeffencePoints << " "; sum+=randomDeffencePoints; }	defense will be equal to sum of defense1 and defense		
Do similar tests described for Goblin for Reptile (defense)	See golbin	See golbin		
Do similar tests described for Goblin for Blue Man	Similar to Goblin, the different is that, there have to be 3 rolls of dice	Make sure that are value for defense1, defense2 and defense3 and values are between **<u>1-6</u>**		
Make sure that appropriat e class is instantiat ed when you use switch statement	cout statements to display the name, strength_ponts and armor for each character type. Strength will be displayed at the beginning of the battle. Armor will be displayed during the battle. Number of dice roles will be displayed during the battle	Compare values that were displayed to the value provided in the table- see above		
Make sure that	int netAttackPoints=player.attack()-	Values that were denerated by rolling		

resulted attack points are calculated correctly.	enemy.deffence();
	cout << endl<< "Total damage - armor is " << netAttackPoints << " - " << enemy.getarmor();
	netAttackPoints= netAttackPoints - enemy.getarmor();
	cout << " = " << netAttackPoints <<end
		dice (see above mentioned tests) are used in the formula, appropriate armor value is subtracted		

Make sure that strength points do not go up during the battle when armor exceeded netAttack Points.	if(netAttackPoints<0){
	
	netAttackPoints=0;
	}
	There is a cout statement for netstrength
		Strength never go up		

Make sure that resulted strength points are calculated correctly	Cout statement with the net Strength points	At the end of the each battle – each time your press f, the resultant strength is calculated using strength- attack		
Check that winner is announce d when one of the characters runs out of strength_ point	cout statement that print sstrength_points after each attack and defense round.	When health is 0 or less, winner/looser message is displayed		
Check that	cout statement that print	Whoever runs out		

whoever runs out of strength point is labeled as looser	sstrength_points after each attack and defense round.	of strength_point first lost of the game		
Test Goblin/ Goblin combination that its possible to win and its possible to loose for a player	Choose goblin for a player and chose goblin for an enemy	Player can win and player can loose		
Test Goblin/ barbarian combination that its possible to win and its possible to loose for each character	Choose goblin for a player and chose barbarian for an enemy or vise versa	Goblin can win or loose Barbarin can lose or win		
Test Goblin/ reptile combination that its possible to win and its possible	Choose goblin for a player and chose reptile for an enemy or vise versa	Goblin can win or loose reprile can lose or win		

to loose for each character				
Test Goblin/ blue men combination that its possible to win and its possible to loose for each character	Choose goblin for a player and chose blue men for an enemy or vise versa	Goblin can win or loose Blue men can lose or win		
Test barbarian/ reptile combination. Maybe sure that each character can win/loose	Choose barbarian for a player and chose replite for an enemy or vise versa	barbarian can win or loose replite can lose or win		
Test barbarian/ blue combination. Maybe sure that each character can win/loose	Choose barbarian for a player and chose blue man for an enemy or vise versa	barbarian can win or loose blue can lose or win		
Test barbarin/ barbarian combinati	Choose barbarian for a player and chose barbarin for an enemy	Player can win and player can loose		

on that its possible to win and its possible to loose for a player				
Test reptile/blue combination. Maybe sure that each character can win/loose	Choose reptile for a player and chose blue man for an enemy or vise versa	reptile can win or loose blue can lose or win		
Test reprile/ reprile combination that its possible to win and its possible to loose for a player	Choose reptile for a player and chose replite for an enemy	Player can win and player can loose		
Test blue man/ blue man combination that its possible to win and its possible to loose	Choose blue man for a player and chose blue man for an enemy	Player can win and player can loose		

for a player				
Make sure that Mega Men attack and defense functions work as expected				
Test enchanted weapon with each character and make sure that double attack points are generated				
Test the spinach option with each character and make sure that if the spinach is eaten at the beginning of the combat, the attack rolls quarduppl e				
Check				

that each character can use venom and that it causes the expected damage, but the expected number of runs.				
Test Mega men with each character and figure out who can win/loose				