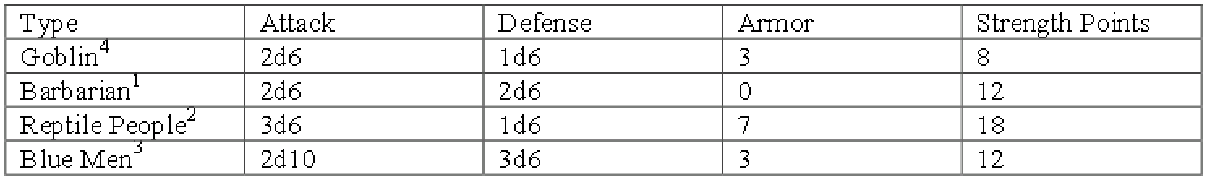
Tatyana Vlaskin
Assignment 3 – Inheritance and Files

Design:

1. First I need to create a parents class, lets call it Character:

```
class Character{
 protected:
        string name;
        int attack,
        int defence;
        int armor;
        int strength_point;
public:
        Character( ); // default constructor
        Character(string &newname, int &armor1, int &strength_point1):
name(newname), armor(armor1), strength_point(strength_point1) {}
        void setAttack(int attack_new);
        void setDefence(int defence_new);
        void setArmor(int armor_new);
        void setStrengthPoint(int strength_new);
        int getAttack();
        int getDefence();
        int getArmor();
        int getStrengthPoint();
        void Wound(int damage, int defence);
```

2. void Wound(int damage, int defence, Character Character1,Character Character2); will look somewhat like this:{
this->Character1.strength_points = this->Character1.strength_points – Character1.getAttack()+Character2.getDefence();} // the set defense function will include the armor component- see below
3. Next step will be to create a subclass for each of the creatures. There are 4 creatures, so I need to create 4 subclasses



Type	Attack	Defense	Armor	Strength Points
Goblin[4]	2d6	1d6	3	8
Barbarian[1]	2d6	2d6	0	12
Reptile People[2]	3d6	1d6	7	18
Blue Men[3]	2d10	3d6	3	12

2d6 is rolling two 6-sided dice. 2d10 is rolling two 10-sided dice.

So subclasses will look somewhat like this:

```
class Goblin : public Character  {
   public:
        Goblin( );
        Goblin(string &newname, int &armor1, int &strength_point1, int attack1, int defence1):
        Character (newname, armor1, strength_point1), attack(SetAttack(attack1)),
        deffence(SetDeffence(defence1)) {}
        GOBLIN: ATTACK 2D6 AND DEFFERNCE 1D6
                voidSetAttack (int attack_new){
                attack1 = 1 + rand % 6;
                attack2 = 1 + rand % 6;
                 attack = attack2+attack2;
                attack_new = attack
                }

        //TOTAL DEFENSE WILL BE CALCULATED using the following formula:
                // Defence = DefenceDiceRolls + Armor Vlue
        void setDeffence(int defence_new){
                deference = 1 + rand % 6;
                //also armor will be added to the defense as well
                defence = deffence +armor;
                defence = defence_new;}

class Barbarian: public Character  {
   public:
        Barbarian( );
        Barbarian(string &newname, int &armor1, int &strength_point1, int attack, int
        deffence): Character (newname, armor1, strength_point1), attack(SetAttack(attack1)),
        deffence(SetDeffence(defence1)) {}
        GOBLIN: ATTACK 2D6 AND DEFFERNCE 2D6
                voidSetAttack (int attack_new){
                attack1 = 1 + rand % 6;
                attack2 = 1 + rand % 6;
                 attack = attack2+attack2;}

        //TOTAL DEFENCE WILL BE CALCULATED using the following formula:
                // Defence = DefenceDiceRolls + Armor Vlue
void setDeffence(int defence_new){
                defence1= 1 + rand % 6;
                defence2= 1 + rand % 6;
                defence = defence2+defence1;
//also armor will be added to the defense as well
                defence = defence +armor;}}
```

<u>class Reptile: public Character {</u>

```
   public:
        Reptile( );
        Reptile(string &newname, int &armor1, int &strength_point1, int attack, int deffence):
        Character (newname, armor1, strength_point1, attack(SetAttack(attack1)),
        deffence(SetDeffence(defence1)) {}

        REPTILE: ATTACK 3D6 AND DEFFERNCE 1D6
                voidSetAttack (int attack_new){
                attack1 = 1 + rand % 6;
                attack2 = 1 + rand % 6;
                attchak3 = 1 + rand % 6;
                 attack = attack1+attack2+attack3;
                }

        void setDeffence(int defence_new){
                defence= 1 + rand % 6;
                //also armor will be added to the defense as well
                defence = deffence +armor;

                }
```

<u>class BlueMan: public Character {</u>

```
   public:
        BlueMan( );
        BlueMan(string &newname, int &armor1, int &strength_point1,int attack, int defence):
        Character (newname, armor1, strength_point1, attack(SetAttack(attack1)),
        deffence(SetDeffence(defence1)) {}

        BLUE MAN: ATTACK 3D6 AND DEFFERNCE 1D6
                voidSetAttack (int attack_new){
                attack1 = 1 + rand % 10;
                attack2 = 1 + rand % 10;
                 attack = attack1+attack2;
                }

        void setDeffence(int defence_new){
                defence1= 1 + rand % 6;
                defence2= 1 + rand % 6;
                defence3= 1 + rand % 6;
                defence = defence1+ defence2+ defence3 + armor;
```

4. When the game starts, the user will be asked what kind of character they want to be:

```
void characterSelection(){
  cout<< endl << "Choose your figher class. " << endl;
  cout<< "1. Goblin." << endl << "2. Barbarian "<<endl << "3. Reptile" << 4. Blue Man " <<
endl;
  int result;
  cin >> result;
  switch(result){
    case 1:
      return Goblin
    case 2:
      return Barbarian
    case 3:
      return Reptile
    case 4:
      return  Blue Man
  }
```

Depending on the selection a corresponding class will be instantiated.

5. Next step, the user will be asked which monster, they want to fight. In real fantasy, it would make sense to randomly chose a monster to fight; however, it will make the testing very difficult and I do not think that I have time for that.
   So the user will be asked which monster they want to fight. There will be a switch statement similar to the one what I provideD above for character selection. Ill place this switch statement into a function and call it, lets say void monsterSelection().
   This function will be called multiple times, inside the

```
void characterSelection() function{
  cout<< endl << "Choose your figher class. " << endl;
  cout<< "1. Goblin." << endl << "2. Barbarian "<<endl << "3. Reptile" << 4. Blue Man "
<< endl;
  int result;
  cin >> result;
  switch(result){
    case 1:
      return Goblin
      monsterSelection()
```

```
        case 2:
           return Barbarian
          monsterSelection()
        case 3:
           return Reptile
          monsterSelection()
        case 4:
          return  Blue Man
          monsterSelection()
      }
```

6. Once we have 2 characters, we start a combat.

Ill make a function like:

```
Void combat(){
while  (Character1. strength_point >0 && Character2. strength_point >0){
            Character1.SetAttack();
            Character2.wound(); // function was described previously
            Character2.SetAttack();
            Character1.wound();
     }
     if (Character2. strength_point<=0){
            cout <<endl<< "Congratulations! You killed the monster!" << endl;}
     if (Character1. strength_point <=0){ // please note that character1 is the user
            cout << "You are dead! You lost." << endl;
   }
}
```

<u>Testing:</u>

For testing I will chose

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	Ill add a cout Statement in each voidSetAttack (int attack_new){ Function and and as the battle prograsses will be check attack point	Each time there is a different attack points		

each time				
Make sure Goblin rolls the dice x2 for attack	Add a cout statement for attack1 and attack 2	Attack 1 and attack 2 have values		
Make sure Goblin gets values from 1-6 for each dice roll	Add a cout statement for attack1 and attack 2	Values of attack1 and attack 2 between **<u>1-6</u>**		
Make sure Goblin attack1 and attach2 value add up and assigned to total attack points	Add cout statement for attack variable	Attack will be equal to sum of attack1 and attack2		
Do similar tests described for Goblin for Barbarian	See Goblin	See goblin		
Do similar tests described for Goblin for Reptile	Similar to Goblin, the different is that, there have to be 3 rolls of dice and dice values are from 1 to 3. So cout attack3	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	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>**		

Man				
Make sure Goblin rolls the dice x1 for defense	Add a cout statement for defense	Defense have value		
Make sure Goblin gets values from 1-6 for each dice roll for defense	Add a cout statement for defense	Values of defense is between **<u>1-6</u>**		
Do similar tests described for Goblin for Barbarian. The difference is that barbarian needs to roll dice x2 for defense	Add a cout statement for defense1 And defense2	Values of defense is between **<u>1-6</u>**		
Make sure barbarian defence1 and defense values add up and to total defense points	Add cout statement for defense variable	defense will be equal to sum of defense1 and defense		
Do similar	See golbin	See golbin		

tests described for Goblin for Reptile (defense)				
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 appropriate class is instantiated when you use switch statement	Add cout statements to display the name, strength_ponts and armor for each character type	Compare values that were displayed to the value provided in the table- see above		
Make sure that wound function works	The wound function calculates strength_points left using the following formulat: Character1.strength_points – Character1.getAttack()+Character2.getDefence();}//please note that armor is part of defense. Put a cout statement for this formular	Values that were denerated by rolling dice (see above mentioned tests) are used in the formula		
Check that winner is announced when one of the characters runs out of strength_point	Add a cout statement that will be print strength_points after each attack and defense round.	When health is 0 or less, winner/looser message is displayed		
Check that whoever runs out	Add a cout statement that will be print strength_points after each attack and defense round.	Whoever runs out of strength_point first lost of the game		

of strength point is labeled as looser				