Suppose that you are creating a game. In this game we have four different types of creatures: humans, cyberdemons, balrogs, and elves. To represent one of these creatures we might define a `Creature` class as follows:

```
class Creature
    {
    private:
        int type;   // 0 human, 1 cyberdemon, 2 balrog, 3 elf
        int strength;   // How much damage we can inflict
        int hitpoints;  // How much damage we can sustain
        string getSpecies(); // Returns type of species, for Creature class
        // returns "unKnown"
    public:
        Creature( );
        // Initialize to human, 10 strength, 10 hit points

        Creature(int newType, int newStrength, int newHit);
        // Initialize creature to new type, strength, hit points

        // Also add appropriate accessor and mutator functions
        // for type, strength, and hit points

        int getDamage();
        // Returns amount of damage this creature
  // inflicts in one round of combat
    };
```

Here is an initial implementation of the `getSpecies( )` function:

```
string Creature::getSpecies()
{
  switch (type)
  {
    case 0:  return "Human";
    case 1:  return "Cyberdemon";
    case 2:  return "Balrog";
    case 3:  return "Elf";
  }
  return "Unknown";
}
```

The `getDamage( )` function outputs and returns the damage this creature can inflict in one round of combat. The rules for calculating the damage are as follows:

- Every creature inflicts damage that is a random number *r*, where *0 < r <= strength*.
- Demons have a 5% chance of inflicting a demonic attack which is an additional 50 damage points. Balrogs and Cyberdemons are demons.
- With a 10% chance elves inflict a magical attack that doubles the normal amount of damage.
- Balrogs are very fast, so they get to attack twice.

An initial implementation of `getDamage( )` is given below:

```
int Creature::getDamage()
{
  int damage;

  // All creatures inflict damage which is a
  // random number up to their strength
  damage = (rand() % strength) + 1;
  cout << getSpecies() << " attacks for " <<
          damage << " points!" << endl;

  // Demons can inflict damage of 50 with a 5% chance
  if ((type = 2) || (type == 1))
     if ((rand() % 100) < 5)
     {
        damage = damage + 50;
        cout << "Demonic attack inflicts 50 "
            << " additional damage points!" << endl;
     }

  // Elves inflict double magical damage with a 10% chance
  if (type == 3)
  {
     if ((rand() % 10)==0)
     {
        cout << "Magical attack inflicts " << damage <<
                " additional damage points!" << endl;
        damage = damage * 2;
     }
  }

  // Balrogs are so fast they get to attack twice
  if (type == 2)
  {
      int damage2 = (rand() % strength) + 1;
      cout << "Balrog speed attack inflicts " << damage2 <<
                " additional damage points!" << endl;
      damage = damage + damage2;
  }
  return damage;
}
```

- Rewrite the class to use inheritance, which will eliminate the need for the variable `type`. The `Creature` class should be the base class.
- The classes `Demon`, `Elf`, and `Human` should be derived from `Creature`.
- The classes `Cyberdemon` and `Balrog` should be derived from `Demon`.
- You will need to rewrite the `getSpecies( )` and `getDamage( )` functions so they are appropriate for each class.

  For example, the `getDamage( )` function in each class should only compute the damage appropriate for that object. The total damage is then calculated by combining the results of `getDamage( )` at each level of the inheritance hierarchy. As an example, invoking `getDamage( )` for a `Balrog` object should invoke `getDamage( )` for the `Demon` object

which should invoke `getDamage( )` for the `Creature` object. This will compute the basic damage that all creatures inflict, followed by the random 5% damage that demons inflict, followed by the double damage that balrogs inflict.

- Also include mutator and accessor functions for the private variables.

Notice that you need to implement getDamage( ) and getSpecies() functions as virtual functions.

- Write a stand-alone function called battleArena you are given the header of it:
  `battleArena(Creature &creature1, Creature &creature2),`
  This function takes two `Creature` objects as input. The function should calculate the damage done by `creature1`, subtract that amount from `creature2`'s hit points, and vice versa. At the end of each round, if one creature has positive hit points but the other does not then the battle is over. This means that the function has a loop until the battle is either a tie or over.
  If both creatures end up with zero or less hit points then the battle is a tie. Otherwise, whoever has hit points less than zero loses.

  Here is a sample main function:

```
int main()
{
	srand(static_cast<int>(time(NULL)));

	Human human1(30, 10);
	human1.getDamage();
	cout << endl;

	Elf elf1;
	elf1.getDamage();
	cout << endl;

	Balrog balrog1(50, 50);;
	balrog1.getDamage();
	cout << endl;

	Cyberdemon cdemon(30, 40);
	cdemon.getDamage();
	cout << endl;

	Elf elf2(50, 50);
	Balrog balrog2(50, 50);
	cout << endl;

	battleArena(elf2, balrog2);
}
```