Objective:

- It will help you understand the benefits of generalization.

Challenge – 1: **(?)**

Suppose that you are creating a fantasy role-playing 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
{
    int strength;   // How much damage we can inflict
    int hitpoints;  // How much damage we can sustain
    int type;       // 0 human, 1 cyberdemon, 2 balrog, 3 elf
    String getSpecies() const; //returns the creature type
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
    //GETTERS
    int getDamage() const;
    int getStrength() const;
    int getHitpoints() const;
    int getType()const;
    //SETTERS
    void setStrength(const int newStrength);
    void setHitpoints(const int newHit);
    void setType(const int newType);
};
```

Here is an implementation of the getSpecies() function:

```
String Creature::getSpecies() const
{
    switch (type)
    {
        case 0:
            return "Human";
        case 1:
            return "Cyberdemon";
        case 2:
            return "Balrog";
        case 3:
            return "Elf";
        default:
            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.
- Elves have a 10% chance to inflict a magical attack that doubles the normal amount of damage.
- Balrogs are very fast, so they get to attack twice.

A skeleton of getDamage() is given here:

```
int Creature::getDamage( ) const
{
    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!" << '\n';
    // Demons can inflict damage of 50 with a 5% chance
    if ((type = 2) || (type == 1))
```

```
        {
            //...
        }
        // Elves inflict double magical damage with a 10% chance
        if (type == 3)
        {
            //...
        }
        // Balrogs are so fast they get to attack twice
        if (type == 2)
        {
            //...
        }
        return damage;
}
```

One problem with this implementation is that it is unwieldy/difficult to add new creatures. 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. Write a `main` function that contains a driver to test your classes. It should create an object for each type of creature and repeatedly outputs the results of `getDamage()`.