Homework 02 – Class Inheritance

Clash of the titans

In this exercise, you are going to have dragons and hydras fight each other.

2.1. Description

Complete the source code according to the instructions below.
WARNING: you should modify neither the beginning nor the end of the provided file. It's thus mandatory to proceed as follows:

1. Copy to your PC the archive. Extract the project from the archive; it should contain everything you need to start coding.
2. Write your code that will make the application work with given code.
3. Save and test your program to be sure that it works properly; try for instance the values used in the example given below;
4. Archive and send the project for review.

The provided main program below simulates a combat between a dragon and a hydra. The hierarchy of classes that models the creatures of this game are missing and you are asked to provide them.
A possible execution example is provided at the end of this document.

2.1.1 The class Creature

A creature is characterized by:

- name (**name**, a constant string);
- level (**level**, an integer);
- number of health points (**lifePoints**, an integer);
- force (**force**, an integer);
- position (**position**, also an integer; for simplicity, our game takes place in 1D (one dimension)).

You must use strictly these attribute names. The attributes must be accessible to classes deriving from Creature.
The methods defined for this class are:

- a constructor allowing the initialization of the name, level, health points, force and position of the creature using the values passed as parameters, in this order; the constructor accepts zero as default value for the position;
- a method **bool isAlive()** returning true if the creature is alive (number of health points greater than zero) or false otherwise;

- a method **attackPoints** returning the number of attack points that can be inflicted by the creature to others; the value is computed as the level multiplied by the force if the creature is alive, or zero otherwise;
- a method **move(int)**, which does not return anything and adds the integer passed as parameter to the position of the creature;
- a method **isDead()**, which does not return anything and displays the message (<name> is no more!) using strictly this format. <name> is the name of the creature;
- a method weaken(), which does not return anything and substracts the number of points passed as parameter from the number of health points of the creature, if it is alive; if the creature dies, its number of health points is set to zero and the method isDead is called;
- a method **showData(),** which does not return anything and displays information about the creature using strictly the following format:

  <name>, Level: <level>, Life points: <points>, force: <force>, \
  Attack points: <attackPoints>, position: <position>

The character '\' is not part of the output and is not followed by a newline.
<name> is the name of the creature, <level> is its level, <points> is its number of health points, <force> is its force, <attackPoints> is its number of attack points and <position> is its position.

2.1.2 The class Dragon

A Dragon is a Creature. It has specific characteristic like the range of its flame (flameRange, an integer). Its specific methods are:

- a constructor which initializes its name, level, number of health points, the force, the range of the flame and the position of the dragon using the values passed as parameters, in this order; the constructor accepts zero as default value for the position;
- a method **fly(int pos)** which does not return anything and allows the dragon to move to the given position pos;
- a method **breathOn(Creature& enemy)** which does not return anything and simulates what happens when the dragon blows its flame towards another Creature:
  1. If the dragon and the creature are both alive and if the creature is in range of its flame, the dragon inflicts its attack points as damage to the creature; the creature weakens by the number of attack points;
  2. The dragon also weakens; it loses n health points, with n being the distance between the dragon and the creature (the further the dragon has to blow, the more it weakens);
  3. If after this epic fight the dragon is still alive and the creature dies, the dragon increases in level by one unit;

The creature is in the range of the flame of the dragon if the distance between them is smaller or equal to the range of the flame (you should use the function distance we provide).

2.1.3 The class Hydre

A Hydre is a Creature. It has as specific characteristics the length of its neck (neckLength, an integer) and the dose of poison it can inject in an attack (poisonDose, an integer). Its specific methods are:

- a constructor which initializes its name, level, number of health points, force, the length of its neck, the poison dose and the position using the values passed as parameters, in this order; the constructor accepts zero as default value for the position;
- a method **injectPoison(Creature& enemy)** which does not return anything and simulates what happens when the hydra poisons another Creature:
  1. If the hydra and the creature are alive and the creature is in range of the head of the hydra, then the hydra inflicts damage to the creature; the creature weakens by the number of attack points of the hydra plus its dose of poison;
  2. The creature is "in range of the head of the hydra" if the distance the creature and the hydra is smaller or equal to the length of the neck of the hydra (you should use the function distance we provide).
  3. If at the end of the fight the creature is no longer alive, the hydra increases in level by one unit;

The function fight takes as parameters a dragon and a hydra. It allows:

- the hydra to poison the dragon;
- and the dragon to blow on the hydra.

2.2 Execution examples

The example of output below corresponds to the provided program.

```
Red Dragon, level: 2, life points: 10, force: 3, attack points: 6, position: 0
ready to battle with:
Bad Hydra, level: 2, life points: 10, force: 1, attack points: 2, position: 42

1st Battle:
creatures are not in range, they can't fight yet.

After battle:
Red Dragon, level: 2, life points: 10, force: 3, attack points: 6, position: 0
Bad Hydra, level: 2, life points: 10, force: 1, attack points: 2, position: 42
Le dragon vole à proximité de l'hydre:
Red Dragon, level: 2, life points: 10, force: 3, attack points: 6, position: 41
Hydra get back one step:
Bad Hydra, level: 2, life points: 10, force: 1, attack points: 2, position: 43


2nd Battle:
+ Hydra attack Dragon with a 3 points attack [ level (2) * force (1) + poison (1) = 3 ];
```

+ Dragon attack Hydra with an 6 points attack [level (2) * force (3) = 6];
+ during his attack, the dragon loses 2 life points
[correspondent to the distance between Dragon and Hydra: 43 - 41 = 2].

After battle:
Red Dragon, level: 2, life points: 5, force: 3, attack points: 6, position: 41
Bad Hydra, level: 2, life points: 4, force: 1, attack points: 2, position: 43
Dragon advance one step:
Red Dragon, level: 2, life points: 5, force: 3, attack points: 6, position: 42

3rd Battle:
+ Hydra attack Dragon with a 3 points attack [level (2) * force (1) + poison (1) = 3];
+ the Dragon attack Hydra with a 6 points attack [level (2) * force (3) = 6];
+ during attack the Dragon loses 1 life point [correspondent to the distance between Dragon and Hydra: 43 - 42 = 1];
+ Hydra is defeated and the Dragon upgrade to level 3.
Bad Hydra is dead!

After battle:
Red Dragon, level: 3, life points: 1, force: 3, attack points: 9, position: 42
Bad Hydra, level: 2, life points: 0, force: 1, attack points: 0, position: 43

4th Combat:
when a creature is beaten, nothing happen.

After battle:
Red Dragon, level: 3, life points: 1, force: 3, attack points: 9, position: 42
Bad Hydra, level: 2, life points: 0, force: 1, attack points: 0, position: 43

The character '\' is not part of the output and is not followed by a newline. The character '\' is used when an instruction have to be written on multi-line, in order to work properly it must be the last character on a line of code.
Reference: MSDN C++ String Literals