Computer Programming 2 1st Test November 14th 2016

·You have 1 hour to complete the assignment.
· If the code does not compile, the exercise won't be accepted for submission.
· Tests are provided to check if your code is correct. **You cannot modify the tests!!**
· If a test does not pass, that exercise won't be accepted for submission.
· Code is expected to be readable, clean, const-correct and optimal.
· A skeleton of the exercise is provided. Feel free to add more files and/or include more libraries if you need them.
· To check if your code is correct, you can write the code you need in the **main()** of the file *test1.cpp*, but leave it **empty before submitting the exam!**
· Inside the code, replace **"INSERT YOUR NAME HERE"** with your name and last name.
· When you finish, *ZIP the whole folder* with a filename called **"lastname_name.zip"** and upload it to "**Test 1"** folder.

We want to create the videogame "*Wars of the 19th Century*", where the players will have the opportunity of commanding the armies confronted in different scenarios like the *Napoleonic Wars* or the *American Civil War*.

Type of units	**Combat value of a single unit (CVU)**
Soldier with bayonet	1
Soldier with musket	2
Light chivalry's horse rider	3
Heavy chivalry's horse rider	4
Mortar	5
Cannon	6

In order to manage the different units of the army, you have to implement:

- The abstract class ***Battalion*** with the following members:
  - string ***name*** as a protected attribute.
  - Public methods:
    - **Constructor** that given a string ***name*** as a parameter initializes the attribute *name* with this value.
    - **getName()** that returns the name of the battalion.
    - **combatValue()** that returns an integer value. It can not modify the attributes of the class or the attributes of the derived classes. This must be the ***pure virtual method***.

- The class ***Infantry*** derived from *Battalion* with the following members:
  - Two integers ***bayonetSoldiers*** and ***musketSoldiers*** as private attributes.
  - Public methods:
    - **Constructor** that given a string ***name*** and two integers, ***bayonetSoldiers*** and ***musketSoldiers*** initializes the attributes of the class.
    - **combatValue()**, that returns an integer with the combat value of the infantry battalion computed as follows:

      bayonetSoldiers * CVU of a bayonet soldier +
      musketSoldiers ***** CVU of a musket soldier

      The CVUs are indicated in the table of the first page.

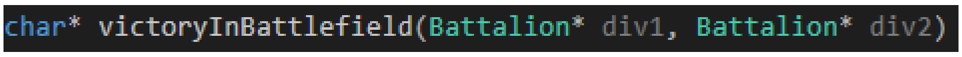
- The class ***Chivalry*** derived from *Battalion* with the following members:
  - Two integers ***lightChivalry*** and ***heavyChivalry*** as private attributes.
  - Public methods:
    - **Constructor** that given a string ***name*** and two integers, ***lightChivalry*** and ***heavyChivalry*** initializes the attributes of the class.
    - **combatValue()**, that returns an integer with the combat value of the chivalry battalion computed as follows:

      lightChivalry * CVU of a light chivalry's horse rider +
      heavyChivalry ***** CVU of a heavy chivalry's horse rider

- The class ***Artillery*** derived from *Battalion* with the following members:
    - Two integers ***mortars*** and ***cannons*** as private attributes.
    - Public methods:
        - **Constructor** that given a string ***name*** and two integers, ***mortars*** and ***cannons*** initializes the attributes of the class.
        - **combatValue()**, that returns an integer with the combat value of the artillery battalion computed as follows:

          mortars * CVU of a single mortar +
          cannons ***** CVU of a single cannon

- Implement the function ***victoryInBattelfield(…)*** that given two *Battalion* pointers returns the name of the Battalion with higher combat value (the one that would win the battle). In the case that both battalions have the same combat values, the function must return the string ***"Same combat value".***



```
char* victoryInBattlefield(Battalion* div1, Battalion* div2)
```