Implement in C/C++ an application to help the local bike store to manage its stock. Implement the followings:

**a)** Keep your shoulders back and focus for the next 1h 30m. (**1p**)

**b)** Represent each bike by an ID, BRAND, YEAR, PRICE, TYPE (Roads, Off-Roads, Touring). The ID should be unique and should never change. (**1p**)

**c)** Show all the bikes having the brand containing a given substring, sorted by YEAR. (**1p**)

(ex. Substring: **al,** shows: (1, **Al**chemy Bikes, 2002, *Touring*), (2, C**al**fee Design,2010, *Roads*), (3, **Al**l-City,2018, *Roads*).

**d)** If no substring is given by the user, at the point **c)**, print the entire list. (**1p**)

**e)** The application should be able to work with 2 extra sub-entities:

* ElectricBike: having the extra field **autonomyHours: int** (**0.5p**)
* BMXBike: having the extra field **wheelsDimension: int** (**0.5p**)

**f)** Update a bike, given the ID, you should be able to change the brand, price, and autonomyHours (for ElectricBike) and wheelsDimension (for BMXBike) (**1p**)

**g)** When the program startpre-insert in your in-memory list 3 “base” bikes, 1 “ElectricBike” and 1 “BMXBike” (**1p**)

**h)** Show only the ElectricBikes (**0.5p**) or BMXBikes(**0.5p**)

**i)** The application is constructed from a modular perspective, using classes. (**1p**)

**j)** Each module has well-defined responsibilities. (**1p**)

**k)** GUI (**extra** **1p**)

**Grading**

Ways to lose points (max 1 point), even if your program works perfectly:

* Your class ignores encapsulation, for example you define everything as public. **(- 0.2 p)**
* You don’t use meaningful names for your class members and methods: for example, your methods are called void m1(), void m2(char \* s) **(-0.2 p)**
* You don’t use indentation and you don’t separate the implementation into a *header* and a *source* file. For example you write the class MyString in a header file on a single line. **(-0.2 p)**
* You don`t mark the required methods as *const* (for example the getters) in their declaration, to ensure that you cannot change any member data through that method. **(-0.2 p)**
* You don’t use a consistent coding style. **(-0.2p)**

**If your project does not compile, it will not be graded!**

**As a rule of thumb, don’t write more than 10 lines of code without compiling your project.**

**If your project does not compile, fix the errors and only when your project is error free you should move on to the next point.**