Implement in C/C++ an application to help the human resource **HR** department from an IT company. Your application should provide the following functionalities:

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

**b)** Represent each employee by an ID, NAME, BIRTH\_YEAR, POSITION (manager, developer, tester). The ID should be unique and should never change. (**1p**)

**c)** Show all the employees having the name containing a given substring sorted by BIRTH\_YEAR. (you should define a method **toString**(), which will return the desired string) (**1p**)

(ex. Substring: **am,** shows: (1,D**am**ian,1982, developer, Intern, 3), (2,J**am**es,1990, tester), (3,Willi**am**,1995, manager, PartTime, 4).

**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:

* InternEmployee: having the extra field **internshipDuration: int** (**0.5p**)
* PartTimeEmployee: having the extra field **workingHour: int** (**0.5p**)

**f)** Add a new employee, the user should decide the type (base class, InternEmployee or PartTimeEmployee) (**1p**)

**g)** When the program start,pre-insert in your in-memory list 3 “base” employees, 1 “InternEmployee” and 1 “PartTimeEmployee”. You will have a **single** vector for all entities. (**1p**)

**h)** Override the toString() in the classes InternEmployee (**0.5p**) and PartTimeEmployee (**0.5p**). The method should concatenate (and return) the base class toString() and the extra field (internshipDuration, respectively workingHour) and the string “**Intern**” respectively “**PartTime**”

**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 Employee in a header file. **(-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.**