TECNICHE DI PROGRAMMAZIONE
ASSIGNMENT 2

1. Introduction

As the semester starts, the First Sub-Committee of the Faculty of Applied Stuff of the *H. and M. Simpson University* faces the task of assigning lecture rooms to lecturers. They ask the local department of Computer Stuff to design *something* that could help them in the task of allocating rooms. The local department first says that "something" is actually a Data Structure ("a what?", they reply) and then gives you the following assignment.

2. Assumptions

To make the problem easier to solve we make the following assumptions:

(1) the set of rooms is not very dynamic; that is, rooms are usually added during the start-up phase of the system and once added a room is never removed. Moreover, we will not perform search operations on rooms (like, "let me see if there is a room named `Aula 21`") and the number of rooms is very small compared to the number of possible time intervals.
(2) once a lecturer has been assigned rooms for his lectures, he/she cannot change his/her mind and ask for different rooms.
(3) we expect that the system will perform several queries in which available of rooms is checked.
(4) each room is identified by a string (use `std::string`);
(5) each room has 10 slots numbered from 0 to 9;
(6) each lecturer is identified by a string (use `std::string`);
(7) a lecturer *request* consists of a sequence of *blocks*;
(8) a block consists of consecutive slots;
(9) an *assignment* of a request consists in assigning each block to a room;
(10) an assignment is compatible with previous assignments if blocks assigned to the same room do not overlap;
(11) assume that all inputs are correct.

3. Specification

Design a data structure `Semester` that supports the following operations:

(1) `AddRoom`: it takes two arguments
   (a) `room`, a string describing a room;
   (b) `system`, the current system;

   and adds the new room to the `system`. All slots of a newly added room are available.

(2) `CheckAvailability`: it takes one block and, for each room, it returns the name of the lecturer that has been assigned that room for an interval that overlaps the block; if the room is available for that interval, nothing is returned;
(3) `AssignRoom`: it takes a *request* from a lecturer and assigns rooms to all blocks of the request for which there is a room. It returns the list of blocks that could not be assigned because all rooms were already allocated.
(4) when we try to print an object of type *Semester*, it prints, for each room, the assigned slots and the name of the lecturer to which the slot was assigned.