# Computer Science – Object Programming in C++

# Laboratory #7

Your task for today is to build a University's Library, that will manage student's books, allow to borrow and return them. Moreover the Student Card is also available and used in the solution.

After implementation of each stage uncomment appropriate section in Main.cpp file to test your code.

The example output from the application is presented in *Example.txt* file.

STAGE_1 (2 Points) – In this stage you have to implement couple of things inside *StudentCard* class. Such as *Constructors* `StudentCard` and `operator<<` (see sample output for more details).

- Constructor which sets *studentID* based on static field `STUDENT_CARD_COUNT` which should be increased each time constructor is called,
- Constructor which copies the the existing *StudentCard*. Restrictions:
    - New *StudentCard* has the ID number set to next available (based on static field `STUDENT_CARD_COUNT` once again),
    - New *StudentCard* has the same validation state as original one,
    - Current *StudentCard* becomes *Invlid*,
    - All *Books* associated should be copied.

Operator should print *StudentCard* ID with status information (Valid or Not Valid), additionally should print all *Books* assosiated with *StudentCard* (operator for *Book* already implemented). Please look at the *Example.txt* file.

STAGE_2 (1 Point) – This stage consists of implementation of *Library* constructor which takes an array of *Books* (this class is already implemented) with size `MAX_BOOKS` and an array of integers representing amount of given books with the same size.

Additionally implement `operator<<` for *Library* which writes existing in *Library* books to the console.

STAGE_3 (3 Points) – In this stage you have to implement *Student* class constructor, that takes *Surname* and *Name* as array of characters with size `MAX_CHAR`. It should sets *Students* data and initialize new *StudentCard* for student.

*Library* function `void AddStudent(const Student& newStudent)` that adds new *Students* to the *Library*. Use `Library::CURRENT_COUNT` variable to keep track of amount of students. If it is greater or equal to `MAX_STUDENTS` then you should ignor new students. You should store only studentID value in the *Library* (function `unsigned short GetID() const` in *Student* class should be implemented – it returns the *ID* value from *StudentCard* associated with *Student*). You should validate *StudentCard* of each student added to the *Library* (function `void ValidateStudentCard()` in *Student* class should be implemented – it sets state of *StudentCard* to `CARD_STATUS::VALID`).

Additionally implement `operator<<` for the *Student* class. It writes *Surname* and *Name*, additionally *StudenrCard* (implemented in Stage_1).

STAGE_4 (1.5 Point) – In this stage you have to implement `void MakeReservation(Library& library, Student& student, unsigned short bookID, unsigned short amount)` function. Make this function `friend` with *Library* and *Student* classes to gain access to their private fields and methods. It makes a reservation of a book associated with given *bookID* for a given *Student*. If *Students' StudentCard* is invalid or *Student* is not added to given *Library* the proces could not be conducted. This function tries to book as many books as possible (use `void BorrowBook(Book newBook)` function from *Student* class – should also be implemented. It just adds book to *Students' StudentCard*) which means that book is granted if there is more than zero books available and student has less then `MAX_STUDENT_BOOKS` books (function `unsigned short GetBookCount() const` in *StudentCard* should be implemented – it returns the amount of books associated with *Students' StudentCard*) and we already granted less books than student requested.

STAGE_5 (0.5 Point) – In this stage *Student* function `int LostStudentCard()` needs to be implemented. This function creates new *StudentCard* for *Student* if current is valid (use copy constructor implemented in Stage_1).