# *EXERCISE SET 3*

## *PROGRAMMING WORKSHOP I, ACADEMIC YEAR 2019-2020*

**Deadline: 22/12/2019, 23:59**

Read the ENTIRE pronunciation carefully and "draw" your program on paper. Decide what variables you will need, what names you will give them, whether constants are needed and if so for what quantities, what control structures to use for each function, and what additional functions are needed. Each time you complete a stage, confirm that your program is working properly before moving on.

If you need clarification or have problems, send a message to the e-class discussion area. Warning: posting code to eclass is not allowed.

This task can be done in groups of up to 2 people. You don't have to be in a group with the same person you are in the lab with or who did the previous assignment. You can discuss the exercises with your fellow students but you are not allowed to exchange code in any way.

Your exercises will be graded on the following (in no particular order):

- Correctness of calculations
- Correct use of pointers, structs, functions and arrays •
  Effective use of appropriate structures, variables, constants, etc. • General program formatting (alignment, variable and constant names, etc.) • Compliance with specifications • Effective comments

The use of goto, the use of gets and the use of global variables are strictly prohibited.

You can assume that keyboard input will always be given in the correct format.

Output messages ending in ':' have a space character after the ':'.

**Instructions for submitting a paper to Autolab** 1. Add

the full names and AEMs of its members to comments at the beginning of the **hw3.c** file group. Please write comments in English characters ONLY. 2. Make a directory

named **hw3submit** and copy **hw3.c** into it

3. Make a text file named **team.txt** and add to it the full names and AEMs of the team members, even if the team consists of one person. 4. Right-click on the **hw3submit** directory and select

**Compress here as tar.gz.** I will
a file named **hw3submit.tar.gz will be created.**

5. Login to Autolab and select **hw3.**

ÿ If you are a group of two and have not already done so, create a group via **Group Options.**

ÿ Submit **hw3submit.tar.gz .**

## 2 **Create a room schedule**

In this assignment you will write a C program that creates the weekly course schedule for a year of study in a university department. The program maintains a table for courses and a table for rooms and places the lecture of a course in a room based on specific criteria specified below.

In summary, the functions specified by this work are the following:

1. Insert a new course 2. Print the
courses 3. Schedule a lecture of
an existing course 4. Print the schedule of the rooms 5. Delete
a course

Lectures can be held on working days of the week (Monday to Friday) and hours between 10.00am. until 3.00 p.m. The minimum duration of a lecture is 1 hour. **Files** You are provided with the files **libhw3.a**

and
**hw3.h** which you are not allowed to change and the file **hw3.c** to which you will add your code. The **libhw3.a** file contains the implementation of the **print_menu** function that prints the program's menu. The menu has the following options:

a: (add course) add a course c: (print
courses) print saved courses s: (schedule lecture) schedule a
lecture for a course p: (print schedule) print the program r: (remove
course) delete a course **q:** (quit) termination of the
program

To compile your program, make sure the **libhw3.a** file is in the same directory as the **hw3.c, hw3.h** files and write:

**gcc -Wall -g hw3.c -o hw3 -lhw3 -L.**

**Defining program data structures** Define a struct describing a course with

the following fields:

a) course identifier (integer),

b) teacher name (character array of size MAX_NAME_LEN) and

c) number of students enrolled in the course (integer).

The program creates an array of courses of size MAX_COURSES, where the contents of each course are all initialized to zero.

Define a struct describing a classroom with the following fields:

a) hall capacity (integer)

b) weekly room schedule (two-dimensional table of pointers to a lesson, with size equal to the number of hours a room can be open times the number of working days of the week (WORKING_HOURS x WORKING_DAYS).

The program creates an array of rooms of size MAX_ROOMS, where the capacity of each room is initialized to zero, and the pointer array for the room schedule has

### Stage 0 - Reading the rooms information from the command line

The program you will write accepts as arguments a series of integers representing the capacity of each hall in non-decreasing order. For example, for three rooms with capacities of 15, 40 and 15 the program should be called as follows: ./**hw3 15 15 40**

In case the user gives as arguments (a) even one non-positive integer value, or (b) the capacity of the rooms in descending order, or (c) more values than the maximum number of rooms (MAX_ROOMS), the message " **Incorrect command-line arguments!"** followed by a newline character and the program terminates.

As long as there is no problem with the arguments, the hall array should be properly initialized based on the capacities given through the arguments. **Attention:** The storage must be done in the order in which the respective capacities were given (non-descending order).

Then, the **print_menu** function is called repeatedly and the user's choice is read from the program input. At this stage, the program should only handle the **q** option in which case the program simply exits. Any other selection has no effect, and simply repeats printing the menu and reading the user's next selection.

### Stage 1 - Add a new course

Write a function that takes as parameters an array of courses and the ID of a course and returns a pointer to a course. The function searches for the course with the given identifier in the course table and if found returns its address, if not found returns the address of the first empty position in the table, while if not found and there is no empty position in the table returns **NULL .**

Write a second function that takes as parameters an array of courses, the id of a course, the number of students in that course, and the name of the teaching professor, and returns an integer. The function, with the help of the previous function, searches for the given course in the table of courses. If it finds it, it returns 0. If it doesn't, it adds it to the first empty position in the course array and returns 1. If it doesn't find it and there are no empty positions in the course array, it returns -1. Note that at any given time the courses are not necessarily stored in consecutive positions in the table.

Add code to the **main** function so that when the user selects the add course function (choice a) the program prints the message **"professor course students:** reads the name of the teaching professor (which does not contain spaces), the course ID, and the number of students enrolled in that course. Assume that id and student count will always be positive numbers (no need to check if the read was successful). Then it calls the second of the two functions above to add a new course to table of courses Depending on the return value of the function, the program prints the message **"C exists"** or the message **"C added"** or the message **"No space",** where **C** is the ID of the course, followed by a newline character.

### Stage 2 - Print the saved lessons

Write a function that takes an array of courses as a parameter and returns **void.** The function prints to the standard output all the courses stored in the table. For each lesson it prints a message of the form **"[C] P S"** followed by a shift character

4 line, where **C** is the unique identifier of the course, **P** is the name of the professor, and **S** is the number of students enrolled in the course.

Add code to the **main** function so that when the user selects the print courses function (choice c) the above function is called with the course table argument.

**Stage 3 - Scheduling a lecture for a course** Write a function

that takes as parameters the table of rooms, a pointer to the course for which we wish to schedule a lecture and the desired duration of the lecture, and returns an integer. The function attempts to place the new course lecture in the classroom table.

He examines the rooms one by one until he finds the first one that can accommodate the number of students in the course. In the room to be examined, he tries to find the first available seat in the course program by checking a) the availability of the room by the hour (starting from 10am) and then by day (starting from Monday) and b) if the course teacher does not he has a lesson at the same time in another room.

If a suitable day/time is found in this room, the lecture is entered in the corresponding slot and the function returns the location of the room in the rooms table. If no suitable seat is found in this hall, the search is repeated for the next largest hall. The function returns -1 if no suitable day/time is found in any room.

Add code to the **main** function so that when the user selects the lecture scheduling function (option s) for a course, the program prints the message **"course** and reads the course ID and the duration of the **duration"** " lecture. If the duration : of the lecture is less than of 1, prints the message **"Invalid** followed by a newline character. If the course does not exist, prints the message **"C not found"** followed by a newline character, where **C** is the course ID. Otherwise, it calls the above function to to schedule the lecture and in case of success it prints **"C scheduled in R"** where **C** is the ID of the course and **R** is the position (in the room table) of the room in which the lecture was registered for this course, while in case of failure it prints **"C not scheduled"** where **C** is the course ID. It then prints a newline character.

**Step 4 - Print the schedule of the rooms** Write a function that

takes as a parameter a pointer to a room, has a return type of **void** and prints the weekly schedule of the room as follows:

Prints a newline character and the string **"Capacity:** X" followed by a newline character, where X is the integer specifying the capacity of the room.

It then prints seven blank characters and whatever days from the string **"MON TUE WED THU FRI"** correspond to WORKING_DAYS followed by a newline character.

For each hour of the day it prints a newline character, the string **"DD:00"** where DD is the integer hour of the day with a range of two digits, and two spaces. Times after 12:00 noon are printed as 1:00 p.m., 2:00 p.m., and so on. For each day of the week, if there is a lecture at that time, it prints the course ID 4 characters wide and left-aligned, otherwise blank, hyphen, and two spaces.

At the end of the weekly program it prints a newline character.

Write a second function that takes as a parameter the hall array, has a return type of **void** and prints the weekly schedule of all halls

using the function above, in the order the rooms are stored in the rooms table.

Add code to the **main** function so that when the user selects the print function of the hall program (option p) the second of the above functions is called.

**Step 5 - Delete Course** Write a function that takes as parameters the course array, a course id and the hall array and returns an integer. The function looks up the course with the given id in the courses table. If the course exists, it removes all of the course's lectures listed in the classroom table, storing the NULL value in each classroom's index table locations. It then deletes the course from the course table. The deletion method is your choice, as long as the contents of the table cells corresponding to already existing courses are not moved. The function returns zero on failure (if the subject in question was not found) or a non-zero integer on success.

Complete the main function so that when the user selects the delete course function (r option) it prints the message **"course:** " (there is a space after the ':' character), reads the course id and calls the above function to delete the course . In case the course is deleted successfully it prints the message **"C deleted",** otherwise it prints the message **"C not deleted"** followed by a newline character, where **C** is the ID of the course.