Middle East Technical University | Department of Computer Engineering

CENG 140

C Programming

Fall' 2021-2022
Take-Home Exam 3

Emre Külah
kulah@ceng.metu.edu.tr
Due date: Jan 30, 2022, Sunday, 23:59

REGISTER.CENG.METU.EDU.TR

1 Overview

Our department collects requests for elective courses every semester. In order to enroll in these courses, we use our credits and make choices. In this assignment, we will implement a program that simulates this system. We will have a student array and a course array. Each student will have a credit amount and will apply to enroll in the courses they want using this credit. Each course has a capacity N_c. After the registration requests are collected, the first N_c students with the highest number of credits will be able to register for the course.

1.1 Student

Each student has an id number and a credit value.

```
struct Student
{
    int id;
    int remaining_bid;
    char* name
};
```

1.2 Course

Each course has an id number, a capacity and a linked list keeping the list of enrollment requests in **descending order** according to the bids given.

```
struct Course
{
    int id;
    int capacity;
    struct Request* requests;
};
```

1.3 Request

This struct keeps a request in a linked list. Each request have a student, a credit value given by the student and a pointer to next request in the list.

```
struct Request
{
    struct Student* student;
    int bid;
    struct Request* next;
};
```

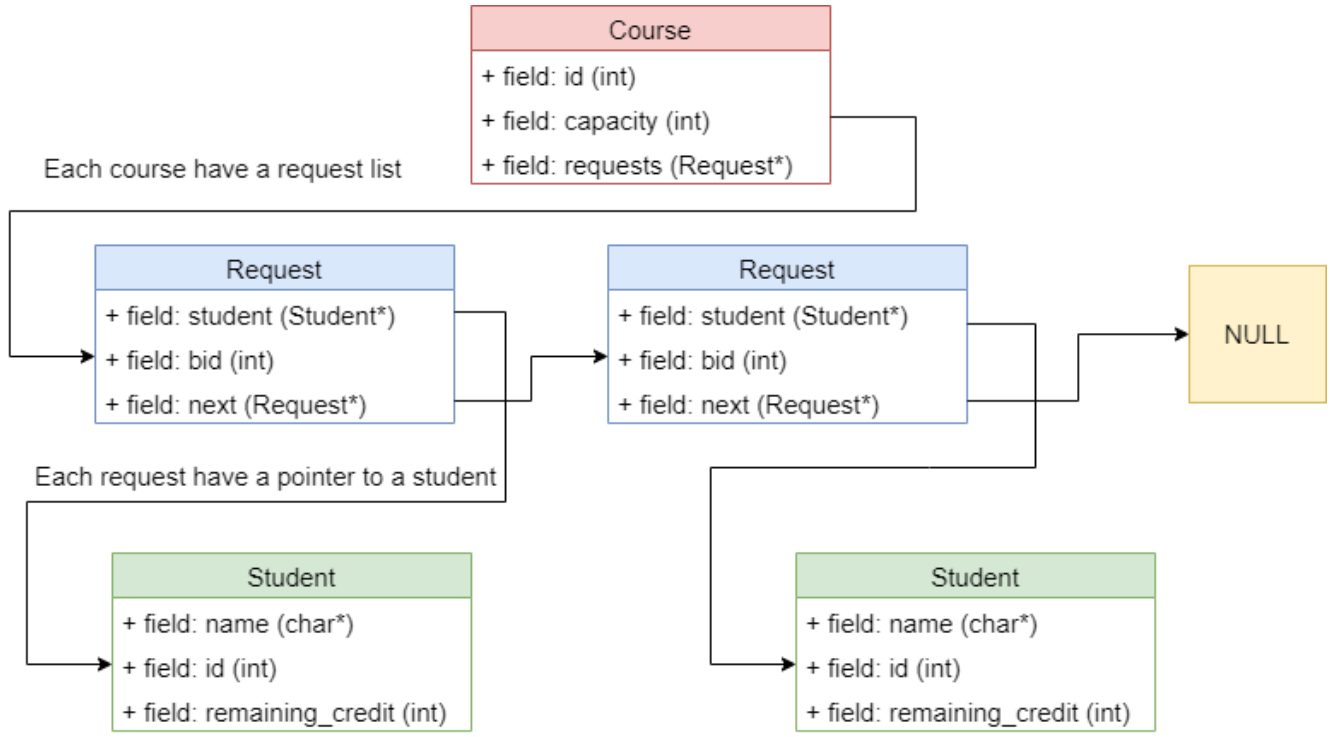

Figure 1: Hierarchy of structs

2 Booking

We will have arrays of courses and students. Each student will apply to some of courses with some bid.

- If student did not apply to that course before and the amount of credit the student gave is less than the student have, then the student will be added to requests list of the course.
- If student applied to that course before, previous request will be removed from the requests list and the credit given in that request will be paid back. Then, new request will be added to the list.
- If the credit the student have does not afford the amount reserved for the request, then request will be rejected and it will not be added to the list.

After the registration collection phase is completed, the list of students who can register for the courses will be printed depending on the capacities.

3 Functions

In this assignment, you will implement 5 functions. One of them will print the request list of a course, two of them will initialize courses and students and two of them will be used for registration.

- **void print_course_requests(Course* courses, int course_count, int stop_at_capacity)**: function gets a course array, size of the array and an integer to check whether it will print the whole requests list or only the ones who are accepted depending on the capacity. If stop_at_capacity = 1, then top N students will be printed for each course with capacity of N. If stop_at_capacity = 0, then all students in the list will be printed. It prints the requests in the following format:

  ```
  Course: 5710140
  Student List: Ahmet(5)
  Course: 5710477
  Student List: Mehmet(7) Ayse(6)
  Course: 5710111
  Student List: Fatma(10)
  ```

  Values in parenthesis represents bid given for that requests.

  If a course has an empty request list then following will be printed:

  ```
  Course: 5710140
  Student List: No students
  ```

  Note: There will be an extra newline character at the end of the student list and there will be **NO** extra whitespace at the end of the print.

- **Student* create_students(int ids[], int remaining_credits[], char* names[], int student_count)**: function gets an id array, an integer array keeping the credit students have, a name array and number of students. It initialize and allocates a student array and create student objects. It returns the array of students.

- **Course* create_courses(int ids[], int capacities[], int course_count)**: function gets an id array and an integer array keeping the capacity course have and number of courses. It initialize and allocates a course array and creates course objects. It returns the array of courses.

- **void new_request(char* student_name, int course_id, int bid, Student* students, Course* courses, int student_count, int course_count)**: function gets a student name, a course id, the bid value student give, students array, courses array, student counts and course counts. It adds the student with given name to request list of the course with given id, if the request meet the requirements. new_request function will check whether request will be added to the list or not or if there is a request belong to corresponding student already, it is going to delete it from the list.

  If student will be added to the list, then the function will print:

  ```
  Ahmet is added to enrollment list of Course 5710111. Remaining credit: 3
  ```

  If student cannot afford the bid decided, then the function will print:

```
Mehmet does not have enough credit to enroll Course 5710477.
```

Note: There will be a newline character after prints. **Note:** You can be sure that student name will be exist in the student list and course id will be exist in the course list. Students names and course ids will be unique. **Note:** You can use strcmp function from string.h to compare two names. Library is already included into the3.c file.

- **void append(Request** requests, Student* student, int bid)**: function gets a pointer to a request list, a student and the bid value student give. It is a helper for new_request function. After new_request function checks whether it will be added to list and calls this function to do appending. **Note:** The descending order of the list must be preserved. **Note:** This function will be tested alone as well. Please do appending operation in this function. **Note:** In case of equal bids, newcomer will be the last of the equals in the list. It will be appended to the end of equal bid requests.

4 Example

I added some extra prints to the example to separate each step. Do not put that kind of prints in your submissions.

We have 3 courses and 4 students. We start with the example main function:

main function

```c
#include <stdio.h>
#include "the3.h"

int main()
{
    int i = 0;
    Course* courses;
    Student* students;

    int ids[] = {5710140, 5710477, 5710111};
    int capacities[] = {3, 5, 7};
    int s_ids[] = {1234567, 1234568, 1234569, 1234570};
    int remaining_credits[] = {7, 8, 9, 10};
    char* names[] = {"Ahmet", "Mehmet", "Ayse", "Fatma"};

    courses = create_courses(ids, capacities, 3);
    students = create_students(s_ids, remaining_credits, names, 4);

    new_request("Ahmet", 5710140, 5, students, courses, 4, 3);
    printf("------------------------------\n");
    new_request("Ahmet", 5710477, 3, students, courses, 4, 3);
    printf("------------------------------\n");
    new_request("Mehmet", 5710477, 2, students, courses, 4, 3);
    printf("------------------------------\n");
    new_request("Ayse", 5710477, 6, students, courses, 4, 3);
    printf("------------------------------\n");
    new_request("Mehmet", 5710477, 7, students, courses, 4, 3);
    printf("------------------------------\n");
```

```
    new_request("Fatma", 5710111, 10, students, courses, 4, 3);
    printf("------------------------------\n");

    print_course_requests(courses, 3, 0);

    return 0;
}
```

Output

```
Ahmet is added to enrollment list of Course 5710140. Remaining credit: 2
------------------------------
Ahmet does not have enough credit to enroll Course 5710477.
------------------------------
Mehmet is added to enrollment list of Course 5710477. Remaining credit: 6
------------------------------
Ayse is added to enrollment list of Course 5710477. Remaining credit: 3
------------------------------
Mehmet is added to enrollment list of Course 5710477. Remaining credit: 1
------------------------------
Fatma is added to enrollment list of Course 5710111. Remaining credit: 0
------------------------------
Course: 5710140
Student List: Ahmet(5)
Course: 5710477
Student List: Mehmet(7) Ayse(6)
Course: 5710111
Student List: Fatma(10)
```

5 Specifications

- In print_course_requests function, there will be **NO** extra whitespace at the end of the list but there will be an extra newline character at the end.
- Do not forget to free stale request which will be changed with a new one.
- Horizontal dashed separators printed in outputs of tests in VPL are printed in test main functions, do not add your own prints for that separators.

6 Regulations

- **Programming Language:** C
- **Libraries and Language Elements:**
  You should not use any library other than *"stdio.h"*, *"stdlib.h"*, *"string.h"*. You can use conditional clauses (switch/if/else if/else), loops (for/while), allocation methods (malloc, calloc, realloc) and strcmp method. **You can NOT use any further elements beyond that (this is for students who repeat the course).** You can define your own helper functions.
- **Compiling and running:**
  DO NOT FORGET! YOU WILL USE ANSI-C STANDARTS. You should be able to compile your codes and run your program with given **Makefile**:

```
>_ make the3
>_ ./the
```

You can run test cases using **Makefile** as well. First, you need to copy the3.h file to the tests folder and run following commands:

```
>_ make testN
>_ ./the
```

testN can be test1, ..., test10.

- **Submission:**
  You will use OdtuClass system for the homework just like Lab Exams. You can use the system as an editor or work locally and upload the source files. Late submission IS NOT allowed, it is not possible to extend the deadline and **please do not ask for any deadline extensions**.

- **Evaluation:** Your codes will be evaluated based on several input files including, but not limited to the test cases given to you as an example. You can check your grade with sample test cases via CengClass system but do not forget it is not your final grade. Your output must give the exact output of the expected outputs. It is your responsibility to check the correctness of the output with the invisible characters. Otherwise, you can not get a grade from that case. If your program gives correct outputs for all cases, you will get 100 points.

- **Cheating: We have zero-tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations and will get 0. Sharing code between each other or using third party code is strictly forbidden. Even if you take a "part" of the code from somewhere/somebody else - this is also cheating. Please be aware that there are "very advanced tools" that detect if two codes are similar. So please do not think you can get away with by changing a code obtained from another source.