

COP 3502- Lab on Dynamic Memory Allocation

In this lab you will be practicing with a dynamic memory allocation coding problem. Then you will be learning how to use the memory leak detector code and check whether your code has any memory leak or not. The steps of using leak detector code in terminal is available in webcourses. For most of your programming assignments in this semester, you will need to use the provided leak detector code. You must try to work on it in the lab with the help of the TA. Doing this will improve your understanding on DMA. Working on this will also better prepare you for the Dynamic Memory Allocation Assignment.

The Coding Problem:

In this problem, you will read a set of students data and their grading information and then process them and then print the requested information.

Problem:

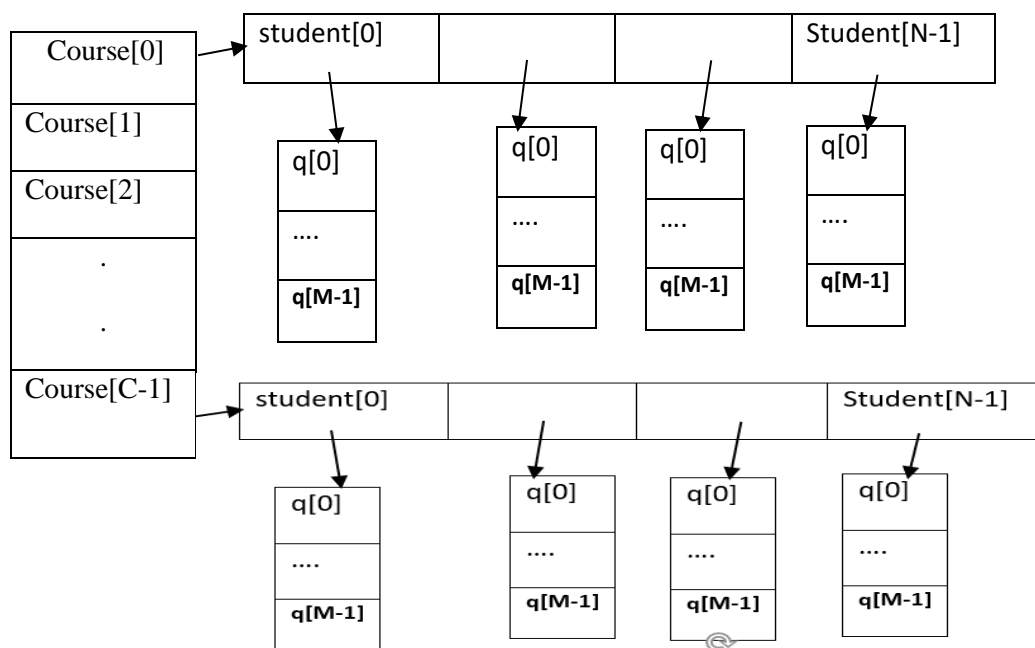
ABC training center offers **C number** of courses. A course has **N number of students**. A student has a last name (single word string with maximum length 50 characters) and a set of course activities, including **one assignment, M number of quizzes** and **a final exam**. The total score of a student is calculated as follows:

The total score = sum of scores from M quizzes + score in the assignment + score in the final exam

The student structure should follow the following definition.

```
typedef struct student
{
    char *lname; //this will require DMA to store a string
    int assignment;
    int finalExam;
    int total;
    int *quizzes;//this will require DMA to store quizzes
}student;
```

You can visualize it by the following picture.



- All the inputs should be taken from standard input (no file I/O operation). Take number of Courses C , number of students N and number of quizzes M as the input from the first line of the. Then Dynamically allocate memory for C courses with N students for each course. For each student, dynamically allocate memory to store scores of M quizzes. Take input for all the scores for quizzes, assignments, and final exams for all the students. Calculate the total scores for each student and store it in the corresponding structure. Do this whole task in a separate method and return the appropriate pointer.

The function header must look like this:

*student** readCourses(int *C, int *N, int *M);*

Display the student's details who achieved the highest total score across all the courses. You do not have to consider if a student with same name is available in multiple courses.

- After writing the result, you should call the release_memory function to free up all the memory.

The function prototype is:

*void release_memroy(student ** st, int C, int N, int M)*

- You should also use the memory leak detector code as instructed in the webcourses.

Sample Input (should be standard input – No file I/O): The input is structured as follows:

Sample input

3 4 2 /// C N M

adel 10 12 9 45 // last name, assignment score, scores for M number of quizzes, and final exam score

smile 6 8 9 39

mahmud 10 12 10 15

jose 8 11 7 41

adam 10 12 9 45 //second course

smith 6 8 9 39

muhammad 10 12 10 45

jones 8 11 7 41

adil 10 12 9 45 //third course

samuel 6 8 9 39

miguel 5 12 10 40

jerry 8 11 7 41

Sample output (must be standard console output) – No file I/O:

Name: muhammad

Assignment: 10

Quizzes: 12 10

Final exam: 45

Total: 77

Course number: 2

The main function of your code should be the following:

```
int main()
{
    atexit(report_mem_leak); //for memory leak detector.
    student **courseList;

    int i, j, C, N, M;

    //passing reference of C, N, and M so that we get to know what we
have in the first line
    courseList = readCourses(&C, &N, &M);

    printHighestTotal(courseList, C, N, M);

    release_memroy(courseList, C, N, M);

    //free(ptr);

    return 0;
}

////////////////////////////////////
```

Use the following command for testing. (Don't put the \$ sign. It is just an indicator that is a command)

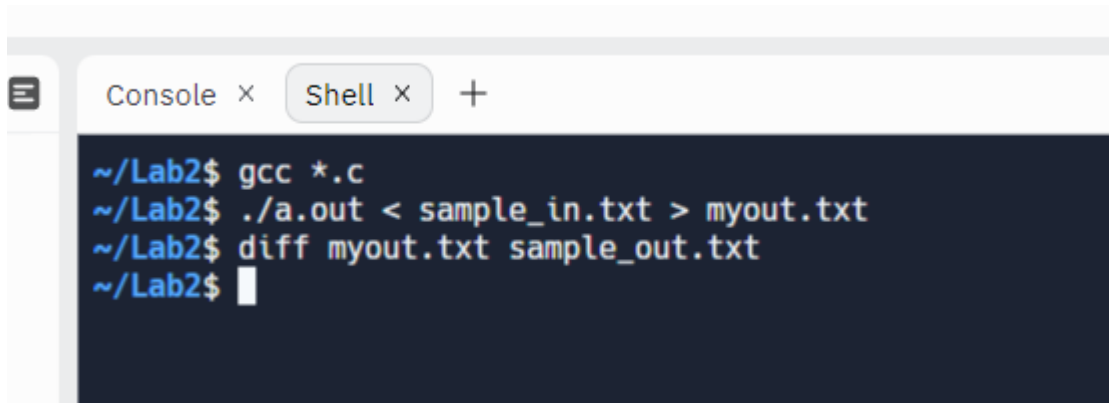
\$gcc main.c //this will compile your c file and generate a.out file as an executable file if your code compiles successfully

\$/a.out < sample_in.txt >out.txt //if there is no error, run your code by this command. It will take input from sample_in.txt file and the output will be generated to out.txt file

You can use various commands to compare your output file against the given sample output file. Note that in your programming assignments you have to exactly match the output with the sample output to get credit.

\$diff out.txt sample_out.txt //the result will be blank if there is no difference between the files
\$ diff -c out.txt sample_out.txt //this command will show ! symbol to the unmatched lines.
\$ diff -y out.txt sample_out.txt //very useful to compare the files side by side to see where exactly it is not matching

A screenshot on how to test your code in terminal:



```
~/Lab2$ gcc *.c
~/Lab2$ ./a.out < sample_in.txt > myout.txt
~/Lab2$ diff myout.txt sample_out.txt
~/Lab2$
```