ENSF 337: Programming Fundamentals for Software and Computer
Lab-5 – Week of October 18, 2021

Department of Electrical & Computer Engineering
University of Calgary

M. Moussavi, PhD, PEng.

In This Lab You Can Work with a Partner (Groups of three or more are NOT allowed):

Working with a partner usually should give you the opportunity to discuss some of details of the topics and learn from each other. Also, it will give you the opportunity to practice one of the popular methods of program development called, **pair-programming**. In this method, which is normally associated with "Agile Software Development" technique, two programmers work together normally on the same workstation (you may consider a zoom session for this purpose). While one partner, the driver, writes the code, the other partner, acts as observer, looks over his/her shoulder making sure the syntax and solution logic is correct. Partners should switch roles frequently in a way that both of them have equivalent opportunity to practice both roles.
If you decided to work with a partner, please submit only one lab report with both names. Submitting two lab reports with the same content will be considered as copies and plagiarism.

Objectives:

This lab consists of several exercises, mostly designed helping you to understand the concept of C `struct` type.

Extended Due Date for All Lab Section (B01, B02, B03, B04):

Based on several requests from students in both lecture sessions, the due date for lab 5 is changed from October 28, to Monday November 1st, 2021. Please notice that lab 6 will be still posted on its regular scheduled date, which is on Friday October 28.

Note:

Lab exercises must be submitted electronically using the D2L Dropbox feature. All of your work should be in a single PDF file that is easy for your TA to read and mark.

Important Notes:

- Some lab exercises may ask you to draw a diagram, that most of the students prefer to hand-draw them. In these cases, you need to scan your diagram with an appropriate device and insert the scanned picture of your diagram into your PDF file (the post-lab report).

- Exercise is not marked, and you shouldn't hand in anything. However, you are strongly encouraged to complete this exercise. Exercises that are not marked are as important as other exercises and during the exams we may have similar questions.

Marking scheme:

The total mark for the exercises in this lab is 31 **marks**

- Exercise A: 8 marks
- Exercise B: 6 marks
- Exercise C: (Not marked)
- Exercise D: 5 marks
- Exercise E: 6 marks
- Exercise F: 6 marks

Exercise A (8 marks): C `struct` Objects on the Computer Memory

Read This First - Structures on the Memory

A structure type in C is a type that specifies the format of a record with one or more members, where each member has a specified name and type. These members are stored on memory in the order that they are

declared in the definition of the structure, and the address of the first member is identical to the address of the structure object itself.
For example, if we consider the following definition for structure `course`:

```
struct course{
      char code[5];
      int number;
      char year[4];
};
```

And, the following declaration of an instance of `struct course`:

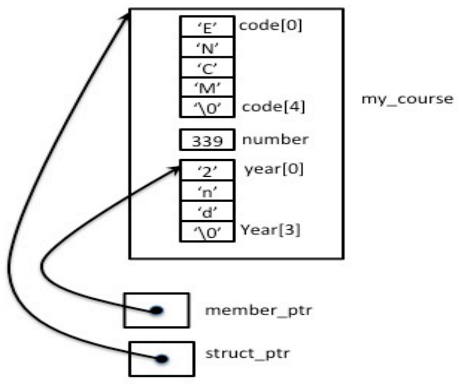
```
struct course my_course = {"ENCM", 339, "2nd"};
```

The address of member `my_course.code` is identical to the address of `my_course`, and the address of member `my_course.number` is greater than the address of the previous member, `my_course.code.` However, the address of the member `my_course.number` will not be necessarily the address of the following byte right after the end of memory space allocated for the member `my_course.code.` It means there might be gaps, or unused bytes between the members. The compiler may align the members of a structure for certain kind of the addresses, such as 32-bit boundaries, to ensure fast access to the members. As a result, the size of an instance of structure such as `course` is not necessarily equal to the sum of the size of the members; it might be greater.

Read This Second – Structures and Pointers

In principle, a pointer to a C structure type is not much different from other types of pointers. They are basically supposed to hold the address of a `struct` object and they are of the same size as other pointers. Please notice, when drawing AR diagrams make sure to be clear whether the arrowhead points to the entire struct instance or to a member of the structure. Please see the following example:

```
      struct course* struct_ptr = & my_course;
      char* member_ptr = mty_course.year;
```



What to Do

Download the file `lab5exA.c`, and `lab5_point.h` from D2L. Read the program carefully and try to predict the output of the program. **Note: when you compile the program, some compilers may display a warning. For this exercise you can ignore this warning.** Now, run the program to compare the results with your prediction. Then, draw an AR diagram for point **one**. Your diagram doesn't need to show the string constants used within printf functions, on the static storage.

What to Submit:

Submit your AR diagram as part of your lab report.

Exercise B (6 marks): Nested Structure

Download file `lab5exB.c` and `lab5exB.h` from D2L. In the file `lab5exB.h` there are two structures called Time and Date and a third structure called Timestamp that nests the other two structures. Study the files to understand what the program does. Then draw memory diagrams for point one in the file `lab5exB.c`.

What to Submit:

Submit your AR diagrams as part of your lab report.

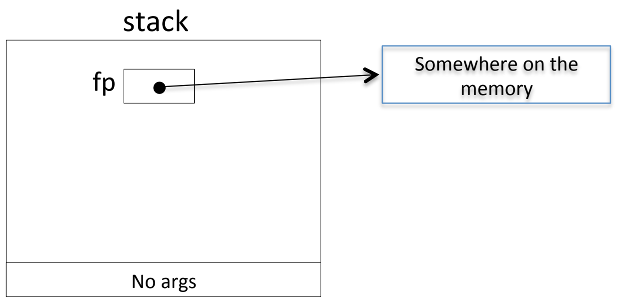
Exercise C (Not marked): Draw AR Diagrams

What to Do:

Download file `lab5exe_C.c, lab5exe_C.h`, and `lab5exe_C.txt,` from D2L. This exercise receives its input from text file, `lab5exe_C.txt`, and populates an array of structure called Bits_Pattern with the available data in this file. Study the content of the downloaded files carefully to understand what the program does. Then draw a memory when the program reaches point one **for the second time**.

Note: in your AR diagram you don't need to show where exactly a FILE pointer points. Just mention it points to "somewhere on the memory" as following figure shows:

```
FILE * fp = fopen("a_file_name", "r");
```



What to Submit:
This exercise will not be marked, and you shouldn't submit anything. You should check your solution with the given solution on the D2L.

Exercise D (5 marks): Writing into a Text File:

What to Do:

Download file `lab5exe_D.c, lab5exe_D.h`, and `lab5exe_D.txt`, from D2L. If you read the given files carefully you will notice that this program reads the content of the file `lab5exe_D.txt` into an array of integers that is declared within the body of a structure called `IntVector`. Then, the program displays the stored values of array on the screen in a single column format. Your task in this exercise is to complete the definition of the function called `display_multiple_column`. Please see the function interface comment for more details.

What to Submit: As part of your lab report submit the printout of the definition of your function `display_multiple_column`, and the printout of the content of the produced text called `lab6exe_D_output.txt`. opened

Exercise E (6 marks): Writing Functions that Use C struct

What to Do:

Step 1: Download file `lab5exE.c` and `lab5exE.h` from D2L, and change the definition of `struct point` to a three dimensional point, by adding the third coordinate of type `double`, called `z`.

Step 2: change the first two lines in the main function to assign values for z-coordinate for struct instances `alpha` and `beta` to 56.0 and 97.0, respectively.

Step 3: modify the definition of any function, if needed, so that they all work for a three-D point.

Step 5: Complete the missing code in functions `distance, mid_point`, and `swap`.

For your information: The distance, `d`, between two three-D point `a(x1, y1, z1)` and point `b(x2, y2, z2)` can be calculated by:

$$d=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$$

What to Submit:

Submit your source code and your program output as part of your lab report (PDF).

Exercise F (6 marks): Using Array of Structures

What to Do:

Download file `lab5exF.c` and `lab5exF.h` from D2L. In this exercise an array of `Point` with 10 elements is created and filled with some sort of random values.

A sample-run of the program shows that the ten elements of `struct_array`, were filled with the following values for <`x, y, z`>. Also, points have labels such as `A9, z9, B7`, and so on.

```
Array of Points contains:
struct_array[0]: A9 <700.00, 840.00, 1050.00>
struct_array[1]: z8 <300.00, 360.00, 450.00>
struct_array[2]: B7 <999.00, 1200.00, 1500.00>
struct_array[3]: y6 <599.00, 719.00, 900.00>
struct_array[4]: C5 <198.00, 239.00, 299.00>
struct_array[5]: x4 <898.00, 1079.00, 1349.00>
struct_array[6]: D3 <497.00, 598.00, 749.00>
struct_array[7]: w2 <97.00, 118.00, 149.00>
struct_array[8]: E1 <796.00, 958.00, 1198.00>
struct_array[9]: v0 <396.00, 477.00, 598.00>
```

Now your job is to write the definition of the functions `search,` and `reverse,` based on the function interfaced comments given in the file `lab5ExF.h`.

What to Submit:

Submit your source code and your program output as part of your lab report (PDF).