Struct and Dynamic Memory
Lab G

TA teaching instructions can be found at the end of this document.

The labs, for this course, are designed to be completed on your own at home or in the 3rd floor Trottier labs. These labs are not graded. You do not hand in these labs. If you prefer to work on a lab with your TA tutorial group, then check the schedule for your TA's tutorial session. You will find this schedule in our MyCourses page under Content/Course Information/TA Information. Since the university has limited lab space, your TA might ask you to bring your laptop and work in a classroom instead of a lab.

This lab is about programming with struct / union and malloc / calloc.

Some labs will have a question zero. These questions will not be covered by the TA during the tutorial. It is extra content meant for you to do on your own.

QUESTION ZERO: Optional problem

If you like, review these online resources:

1. Struct and union: https://www.programiz.com/c-programming/c-structure-examples
2. Dynamic memory: https://www.programiz.com/c-programming/c-dynamic-memory-allocation

QUESTION ONE: Struct and Union

Given the following structure:

```
struct BANK_ACCOUNT {
        char type; // 'S'=savings, 'C'=checking
        double balance;
        union ACCOUNT_SPECIFIC {
                double charge; // for withdrawal from savings accounts
                int credit_score; // for checking account
        } specific;
};
```

Do the following:

a) Create a single source file called bank.c and put into that file the above structure and a main() function.
b) Create a local array, called accounts, in main(), of 100 cells, type BANK_ACCOUNT. Use this pattern: `struct BANK_ACCOUNT accounts[100];`
c) Create a local integer variable, called `nextSpot`, and initialize it to zero. This variable will be used to track where in the array `accounts` the next account will be created. In other words, to the left of `nextSpot` are array cells with account information. At `nextSpot` and to the right, those cells are not used.
d) Prompt the user for the number of accounts they would like to create. Store that in an integer variable called `newAccounts`.

e) Using a for-loop with the `newAccounts` variable, `nextSpot`, and the array `accounts`, prompt the user for the information to input into the bank accounts. You will need to first ask the user the type of account, then the remaining fields depending on the account type.
f) The program ends by displaying all the cells of the array from 0 to `nextSpot`, to confirm that the information was input correctly.

NOTE: Sample code is provided showing one way you might program QUESTION ONE. I suggest you try doing question one on your own before looking at the sample code. The TA will review the sample code during their mentoring time.

QUESTION TWO: Dynamic Memory

Redo Question One, but replace the array with: `struct BANK_ACCOUNT *accounts;`

Your are still building an array, so use calloc() to allocate 100 cells, for example:
`calloc(100,sizeof(struct BANK_ACCOUNT));`

Then modify all the array references to pointer references.

Call your source file Bank2.c.

You have completed your lab.

TA Teaching Instructions

The lab time is divided into two 30-minute periods. The first period is a lecture. The second period is the lab. The lab period is conducted in a TA directed setting where students do the above lab together with the TA. The first period is conducted as a lecture and covers the following material:

- Ask the students if they would like you to review struct and union
- Review the basic syntax of programming with malloc() and calloc() with structures
- Review the example code provided in this lab
- Do this lab with the students