THE UNIVERSITY OF WESTERN ONTARIO

DEPARTMENT OF COMPUTER SCIENCE
LONDON CANADA

Software Tools and Systems Programming

(Computer Science 2211a)

ASSIGNMENT 4

Due date: Wednesday, November 10, 2021
11:55 pm Eastern Standard Time – 4:55 am Greenwich Mean Time)
allow up to one day late ONLY – assignment closed Nov. 11, 2021 11:55pm est : 4:55am GMT

Assignment overview

The purpose of this assignment is to provide the student with experience with pointers and variable addresses in memory. The other purpose of this assignment is to provide the student with more than just experience writing code. This assignment will also concentrate on introducing the good programming practices of modularity and incremental programming.

PREPERATION:
For this assignment, create a new directory under the assignments directory created in the first assignment. Label this new directory: asn4

All work should be performed in this directory. Use a UNIX editor like vi to create and compile the C code.

The code MUST compile in the UNIX environment to be considered correct.

This time the assignment will be contained within multiple files.

Assignment Overview:

The purpose of this assignment is to provide the student with more than just experience writing code. This assignment will also concentrate on introducing the good programming practices of modularity and incremental programming.

When students first start to learn to code, the exercises are fairly simple and small. As such we begin the habit of just starting to write code that completely performs the requested actions and we place most, if not all, of the code in the main function.

The reality is that although this may be adequate for uncomplicated tasks, as the programs become more detailed and complex, this practice will make the performance of programming more and more injurious to the process.

This assignment will aid in the introduction to three good programming practices. These are modularity, incremental coding, and correct use of controls (loops and conditionals).

The core of the assignment is to dynamically allocate and use an array of arrays in the form of an array that holds the addresses (pointers/references) to a collection of two dimensional arrays. The program will prompt the user for the number of arrays and then will prompt for the two dimensions of each array in the collection (rows and columns). The program will then proceed to dynamically allocate the space for these arrays and then randomly populate each individual array with integer values. The program will then combine all the values in the array collection based on element location. This will entail adding all the values in each position of all the arrays in the collection (e.g. [0][0] , [0][1], [0][2], ...) and storing the totals in the corresponding position in the last array of the collection. The program will finish by sorting the last array (that contains the totals) in ascending order. The program will loop and start over by prompting for a new series of arrays.

Regarding modularity, each of these operations will be contained in their own function and each function will reside in their own **.c file**. The main function will be the program control section.
Regarding incremental programming, each operation will be a separate project, where, after the first project, each one will build off the last.

Correct use of controls will be introduced by forbidding the use of infinite loops with conditional break statements imbedded in the control.

Example:

```
while (1) {
     ... c statements
     ... c statements
     ... c statements
      if (something)
           break;
     ... c statements
     ... c statements
}
```

This is tantamount to using a hammer to drive in a screw. Yes, it works, but it is NOT pretty, and it does NOT do the job correctly. This causes the programmer to look and attempt to validate the logic of the loop termination. This practice will result in major deductions on your assignments.

The break and continue statements are powerful and useful parts of the C language when used correctly. Using it in the above example is counter to the use of the while loop. This assignment will contain a possible location creating an example of the correct usage of the break statement in C code (this is optional).

Your main.c must only contain minimal computation (as demonstrated in class).
Below is a start for your main() function. No other functions are to be included in your main.c file.

hint: notice the single `#include` in the main.c

The size array containing the pointers to the arrays in the assignment is created in dynamic memory so it can contain any number of elements.

The declaration for this array is:

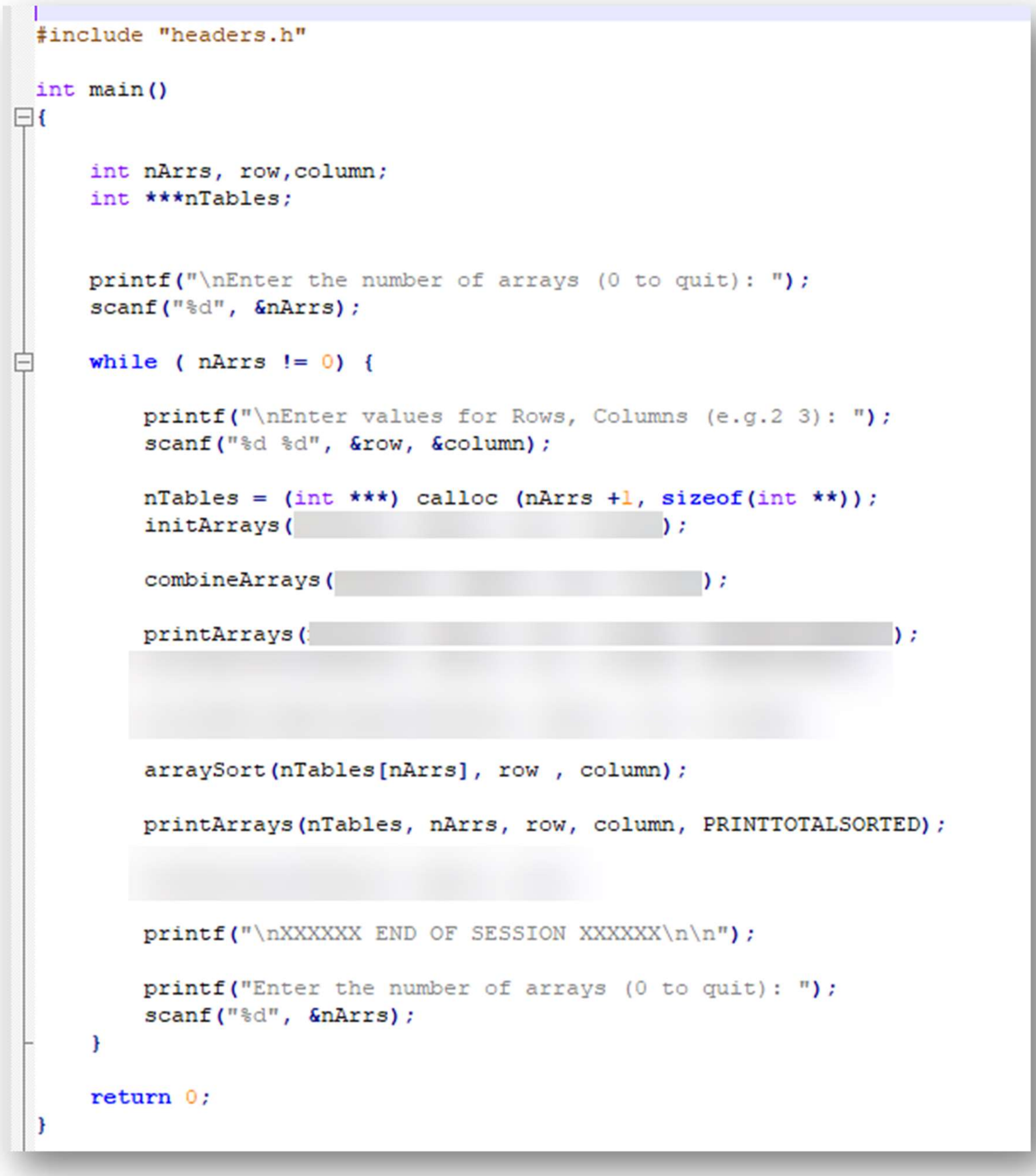
```
int ***nTables;
```

(you are allowed to name the variable any label you wish, but it must be a triple pointer.)

You are to allocate one extra array that will hold the totals of all the arrays. This must be the nth array. (e.g. if the user want four (4) arrays in the collection, then you must allocate for five (5) arrays. The fifth array (nTables[4]) will be set aside to hold the calculated totals.)

The blurred out sections of the main allow the student to develop the code to accomplish the requirements of the assignment as they see best.

main.c

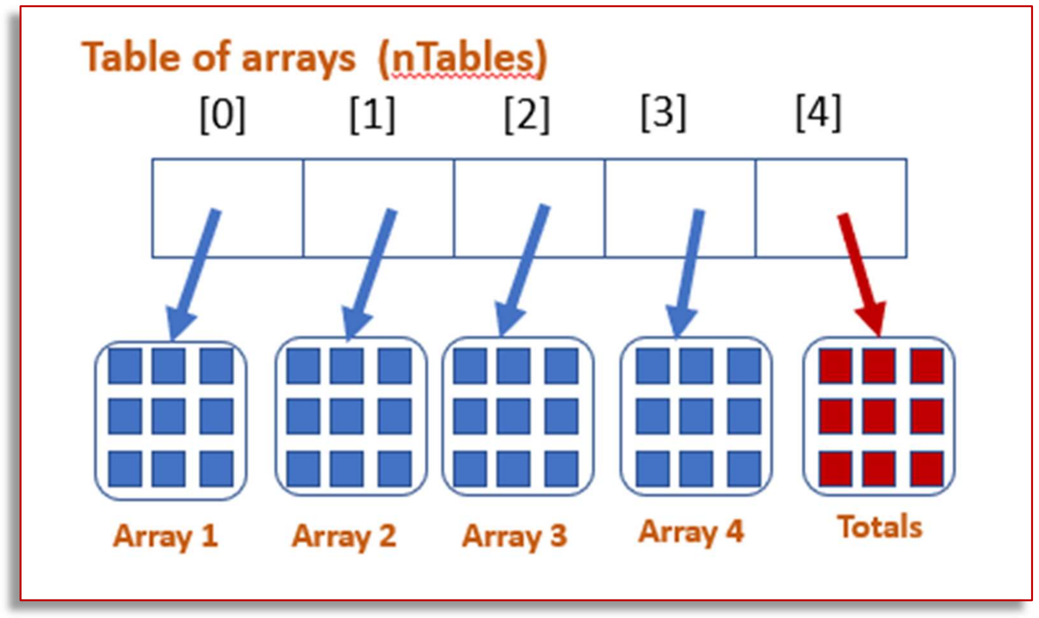


```c
#include "headers.h"

int main()
{

    int nArrs, row,column;
    int ***nTables;


    printf("\nEnter the number of arrays (0 to quit): ");
    scanf("%d", &nArrs);

    while ( nArrs != 0) {

        printf("\nEnter values for Rows, Columns (e.g.2 3): ");
        scanf("%d %d", &row, &column);

        nTables = (int ***) calloc (nArrs +1, sizeof(int **));
        initArrays([redacted]);

        combineArrays([redacted]);

        printArrays([redacted]);
        [redacted]

        arraySort(nTables[nArrs], row , column);

        printArrays(nTables, nArrs, row, column, PRINTTOTALSORTED);
        [redacted]

        printf("\nXXXXXX END OF SESSION XXXXXX\n\n");

        printf("Enter the number of arrays (0 to quit): ");
        scanf("%d", &nArrs);
    }

    return 0;
}
```

This main.c is a suggestion, but illustrated the requirement that the main() function is just a control function. Your main() function can be different as long as it adheres to this concept.

You **MUST** adhere to the standard that little or no computation occurs in the main() function.:

The topology will be an array of two-dimensional arrays:



Create a program that:

1) prompts the user for three values. (You do not have to check for valid input type.)
   First, prompt for the total number of arrays (0 to quit):
      (1st) - the number of arrays in the collection.
   Then prompt for the dimension (rows and columns) of every array in the collection.
      (1st) - the number of rows in each array
      (2nd) - the number of items (places) in each row (i.e. columns)
   The image above is based on the inputs of 4 arrays of three rows and three columns.
2) initializes the arrays in the collection (each two dimensional array):
      create a function called InitArray() and place it in a file labeled InitArray.c
      using the code below for a random number generator
      fill each two dimensional array with random integers between 1 and ten times the total number of elements in the collection (above example – 4 * 3 * 3 * 10) inclusive.
      Do not fill in the last array with random numbers (hint: set each element in the last array to zero (0)
3) prints out the collection of arrays (not including the last array)
      create a function called PrintArray() and place it in a file labeled PrintArray.c
      label the Array Number
      This does not print out the last array.
      PrintArray() should be generic and be used to print out the arrays so the code is reused and not repeated unnecessarily.
4) combine all the arrays in the collection by adding the totals of each element position.
   - add all the values in position [0][0] and store this total in the [0][0] position of the last array in the collection.
   - repeat this for every position [0][1] then [0][2], etc.
   print out the results by displaying the contents of the last array in the collection.
5.) Sort the combined array (the last array only) into ascending order.
   print out the last array again in ascending order.

Code Standards:
1.) create two (2) header files.
one file labeled headers.h with the following contents:

```
#ifndef HEADERS_H_INCLUDED
#define HEADERS_H_INCLUDED

    #include <stdio.h>
    #include <stdlib.h>
    #include <time.h>
    #include "definitions.h"

#endif // HEADERS_H_INCLUDED
```

2.) the other file is labeled definitions.h with the following contents:

```
- any preprocessor definitions (e.g. DEBUG)
- all function prototypes.
```

3.) all arrays are to be dynamically allocated based on user input.

Required Coding Standards
All code is to be indented correctly.
Comments at the very beginning (top – first lines) of each of the C code files must be:

```
/* CS2211a 2021 */
/* Assignment 04 */
/* your name */
/* your student number */
/* your UWO User Name */
/* Date Completed */
```

Your program is to be submitted as C code file.
Your script will be a script file created in UNIX.

All variables MUST have a comment describing their intended use(s) demonstrating an understanding of what each variable is used for.

A comment describing the code for each major part of the code is expected. Comment(s) can be added to describe any complete statements you create. They can be brief but must convey what that section of code performs.

Any questions or ambiguities are to be asked through the Forums only. Any individual emails containing questions on this assignment will be replied to with a request to restate the question in the Forums in Owl where they will be addressed.

Working in UNIX.

Save the files in your asn4 directory.

NEXT: Follow the steps below to complete Part 2.

1. Type the following to begin recording your session in a file called yourUserName_asn4.output

    script YourUserName_asn4.output

    note - (using your actual user name).

2. Display the current date and time using the appropriate command

3. Display your username using the appropriate command

4. Display the contents of the current working directory using the 'l' switch (lower case L).
5. Display the contents of the file main.c (i.e. show the main() function in your C program)

6. Compile the program again ensuring the executable is labeled: **`asn4`**
   (yes, even though we have not covered makefiles, you can use one if you wish)

7. Run the program.

8. Type **`exit`** to stop your screen capture session.

9. Compress the code files and the listed required submission files into a single **.tar.gz** file.
   (i.e. Ensure you have a complete copy of the Asn4_SubmissionForm in your asn4 directory.)

main.c
headers.h
definitions.h
initializeArrays.c
printArrays.c
sortArrays.c
combineArrays.c

note: other function (and files) can be included or added to this list.
this list is the minimum files to be created.
remember: any unit of work should be in separate .c (or .h) files.
example: printArrays.c can contain two or more print functions.
but, unrelated operations like combine totals must be in their own file
*YourUserName*_asn4.output
Asn4_SubmissionForm.txt (or .pdf)

10. Copy (i.e. using an FTP or any method of your choice) the .tar.gz file to your computer so you can upload it through OWL for submission.

Submission Instructions:

Complete the *CS2211 Assignment Submission Form* Name (or rename) that form to **Asn4_SubmissionForm.txt (or Asn4_SubmissionForm.pdf)**

Save this form in your asn4 directory as a text file or as **PDF** (most word processors have this option).

Submit via the CS2211 OWL Web Site the files in your asn4 directory using the instructions on how to compress and submit the single compressed .tar.gz file.

Submit via the CS2211 OWL Web Site the following three files inside your compressed submission file:
(all your program files (.c and .h)
*YourUserName*_asn4.output
Asn4_SubmissionForm.txt (or .pdf)

note: do CAN (optional) include your executable file: asn4 (it is easier than trying to remove it ...)
note: you are submitting one tar file (and only one file).
this is a compressed file that contains the files listed above.
do NOT send any of the three above files separately.
note: If you have elected to use an SRA – it must be denoted in the Asn4_SubmissionForm or the SRA will not be applied.
note: marks will deducted if the Asn4_SubmissionForm is omitted.

It is the student's responsibility to ensure the work was submitted and posted in OWL.
OWL replies with a summation verification email (every time).

Submission date and time is based on the last file submitted. So if you re-submit after the due date, the entire assignment will be graded as late based on that timestamp.

The teaching assistant grading your assignment will compile and run your program.
If the program does not compile under UNIX, the TA will NOT attempt to correct or fix your program so it will run.

(*yourUserName* - example: assume my UWO email is kdoit373@uwo.ca
i.e. if my email is – **kdoit373@uwo.ca** then my user name will be – **kdoit373**
So, my UWO User Name is: **kdoit373** and this assignment is **asn1**
therefore, one of the file names that is to be used for submission is:
kdoit373_asn1.c

It is the student's responsibility to ensure the work was submitted and posted in OWL.
OWL replies with a summation verification email (every time).

Any assignment **not** submitted correctly will **not** be graded.

PS: remember: do your own work – you will need to know all this for the exam to pass !!!!

Please check CS2211 Assignment Submission Guidelines.
note: Guidelines mention the example for assignment 1 – please substitute a 4 for the 1.