Programming and Data Structures Laboratory, 2018-19 Spring semester, Section 6

March 26, 2019: Tutorial and Assignment 8 (Dynamic memory allocation, Structures)

Tutorial (for practice)

Take two integers m and n as input from the user. Dynamically declare (i) a 2-D array of integers *iarr*, and (ii) a 2-D character array *carr*, both of dimension m x n. Print out the address of each element of each of the arrays. Understand how dynamically allocated arrays are stored in the computer memory.

Assignment (for evaluation – write on machine and submit to Moodle before end of class)

Note for all questions: You should use dynamically allocated arrays and local variables only, as specified by the questions. Using global variables will be severely penalized. Also, if a question asks to write a function, writing the whole code within main() will be penalized severely.

1. [20 marks] Define a structure **student_info** having two fields, an integer **namlen** and a character pointer **name**. Take an integer n as input through keyboard. Dynamically allocate an array of n structures of the above type. Each structure in that array will correspond to one of n students. For each student, first take the length of the name of the student as input through the keyboard. Store it in the **namlen** field of the corresponding structure. Then take the name of the student as input through the keyboard, and store it in a dynamically allocated array (of size **namlen**) pointed to by **name**. Finally, sort the student information alphabetically with respect to the names. Display the sorted sequence of names on the screen, with each name in a different line.

2. [20 marks] Take three integers m, n and p as inputs through the keyboard. Assume that m,n,p<10. Dynamically allocate two integer arrays A and B, of dimensions m x n and n x p respectively. Fill both arrays A and B by taking integer inputs through the keyboard. Print both matrices on the screen. Then write a function Mat-mul() to multiply the two matrices A and B with the following prototype:

int *Mat_mul(int *, int *, int, int, int);

After taking the matrices A and B as inputs, **you call the function Mat-mul() exactly once with appropriate arguments.** After the call returns, print the product of matrices A and B computed by Mat-mul() on the screen.

Sample input / output:

Enter m, n and p: 2 4 3

Enter elements of Matrix A:

Enter entry (1,1): 1

Enter entry (1,2): 3

Enter entry (1,3): -2

Enter entry (1,4): 4

Enter entry (2,1): 6

Enter entry (2,2): 0

Enter entry (2,3): 0

Enter entry (2,4): 9

Enter elements of Matrix B:

Enter entry (1,1): 0

Enter entry (1,2): -20

Enter entry (1,3): 2

Enter entry (2,1): 4

Enter entry (2,2): 9

Enter entry (2,3): 0

Enter entry (3,1): 0

Enter entry (3,2): -2

Enter entry (3,3): -3

Enter entry (4,1): -1

Enter entry (4,2): -12

Enter entry (4,3): 20

Matrix A:

1 3 -2 4

6 0 0 9

Matrix B:

0 -20 2

4 9 0

0 -2 -3

-1 -12 20

Product Matrix:

8 -37 88

-9 -228 192

Submission instructions:

Submit one compressed file, named as **<roll number>_A8.tar.gz** or **<roll number>_A8.zip**

The compressed file should contain two source files:

<roll number>_A8_1.c, <roll number>_A8_2.c