

Lab 06 Specification – Getting Hands-on Experience in using Pointers and Dynamic Arrays.

Due (via your git repo) no later than 8 a.m., Friday, 2nd November 2018.

50 points

1 week Individual Lab

Lab Goals

- Practice Dynamic Arrays
- Practice Pointer Arithmetic

You are required to work individually on this lab. Lab work may be done offline. But the expectation is that every student should attend at least the first 20 minutes of the lab sessions. As you may be aware that, I go around and check with the students in the lab after the first 15 minutes of the lab session (that is, around 2:45 PM precisely) for clarification questions on the lab requirement. I expect every student should be in attendance, and be present in the lab till that time. This will also help the student to interact with other students, teaching assistants, and the Professor.

Both the TA's and I will be available to answer your questions during the entire lab period. So, you should take advantage of our time.

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at:
<https://www.cs.allegheeny.edu/sites/amohan/resources/suggestions.pdf>

Learning Assignment

To do well on this assignment, you should also read

- **KR** - chapter 05: [5.6-.5.9];
- Class discussion notes and slides.

Assignment Details

In this lab assignment, you will work on creating dynamic arrays using pointers. Additionally, you will work on pointer arithmetic.

Part 1: Dynamic Single-Dimensional Array)

Write a program called **onedim.c**, with five different functions called "createDummyArray", "storeIntoArray", "printArray", "findMaxInArray", and "findMinInArray".

1. Implementation details of "createDummyArray" function: First, the function should prompt the user to type in the number of elements in the array. The user input should be stored into a global variable called "size", that would be accessed and used throughout the program by all the different functions. Using "malloc", create an array of elements that is of integer data type. The number of elements in the array, should be equal to the number of elements provided by the user. The function should **return** a pointer of type integer, that hold the base address of the array. Note: A major difference between using a variable length array and dynamic array is that a dynamic array uses the heap memory and thereby one can create a large array by provisioning memory blocks directly from the heap. In contrast, a variable length array uses a stack memory and thereby there is a space restriction on the size of the array. Hence, it is very efficient to create a dynamic array. Also, it is worth making a note that the static array is limited to the size at compile time and cannot grow/shrink at runtime. However, a dynamic array has the ability to grow/shrink at runtime. WOW, isn't that exciting. Using Dynamic arrays, you can implement any data structure like linked list, stacks, and queues, that deals with no knowledge about the size of the array in advance. I expect that you declare the function as follows:

```
int *createDummyArray() {
    ----
    ----
    Your logic goes here
    ----
}
```

2. Implementation details of "storeIntoArray" function: The function takes a pointer of type integer as input. Based on the value in the global variable "size", an iterative loop should be constructed that would capture all the different values to be filled inside the array based on the user input. As the user type in the different values to be stored in the array, the values are then stored in the array. The function does not return anything, and hence the return type should be void. I expect that you declare the function as follows:

```
void storeIntoArray(int *ptr) {
    ----
    ----
    Your logic goes here
    ----
}
```

3. Implementation details of "printArray" function: The function takes a pointer of type integer as input. Based on the value in the global variable "size", an iterative loop should be constructed that would simply print all the values (tab separated in one line) stored in the array. The function does not return anything, and hence the return type should be void. I expect that you declare the function as follows:

```
void printArray(int *ptr) {
    ----
    ----
    Your logic goes here
    ----
}
```

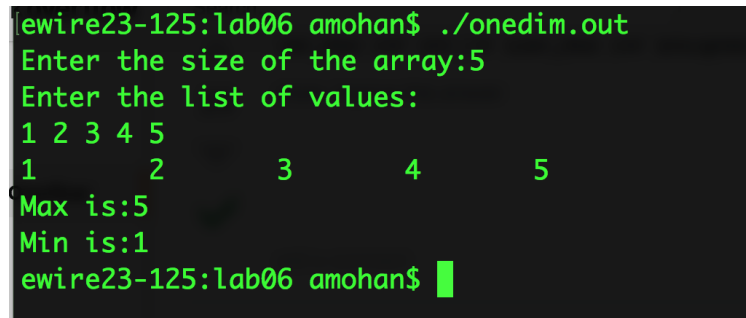
4. Implementation details of "findMaxInArray" function: The function takes a pointer of type integer as input. Based on the value in the global variable "size", an iterative loop should be constructed that would loop through all the elements in the array and find the maximum value stored in the array. The function should return the maximum value of type integer. I expect that you declare the function as follows:

```
int findMaxInArray(int *ptr) {
    ----
    ----
    Your logic goes here
    ----
}
```

5. Implementation details of "findMinInArray" function: The function takes a pointer of type integer as input. Based on the value in the global variable "size", an iterative loop should be constructed that would loop through all the elements in the array and find the minimum value stored in the array. The function should return the minimum value of type integer. I expect that you declare the function as follows:

```
int findMinInArray(int *ptr){  
    ----  
    ----  
    Your logic goes here  
    ----  
}
```

A sample screenshot of the expected output is as follows:



```
ewire23-125:lab06 amohan$ ./onedim.out  
Enter the size of the array:5  
Enter the list of values:  
1 2 3 4 5  
1      2      3      4      5  
Max is:5  
Min is:1  
ewire23-125:lab06 amohan$
```

In order to get a maximum number in C, you may need to use the standard library called limits.h and the following code:

```
int maximum = INT_MAX;
```

The screenshot shown above is just for a sample. Your program should produce the correct result based on the user input values.

I expect that you would create a main function, that would orchestrate all the above 5 function calls, to produce the output that matches the expected output.

Part 2: Dynamic Two-Dimensional Array)

Write a program called **twodim.c**, with seven different functions called "createDummyArray", "storeIntoArray", "printArray", "findMaxInRows", "findMinInRows", "findMaxInCols", "findMinInCols".

1. Implementation details of "createDummyArray" function: First, the function should prompt the user to type in the number of rows and cols used in constructing the two dimensional array. The user input should be stored into global variables called "rowsize" and "colsize", that would be accessed and used throughout the program by all the different functions. Using "malloc", create an array (2D array) of elements that is of integer data type. The number of elements in the array, should be equal to (rowsize * colsize), which is computed based on the user inputs. The function should **return** a pointer of type integer, that hold the base address of the array. I expect that you declare the function as follows:

```
int *createDummyArray() {
    ----
    ----
    Your logic goes here
    ----
}
```

2. Implementation details of "storeIntoArray" function: The function takes a pointer of type integer as input. Based on the value stored in the global variables "rowsize" and "colsize", an iterative loop [nested for] should be constructed that captures all the different values to be filled inside the array based on the user input. As the user type in the different values to be stored in the array, the values are then stored in the array. The function does not return anything, and hence the return type should be void. I expect that you declare the function as follows:

```
void storeIntoArray(int *ptr) {
    ----
    ----
    Your logic goes here
    ----
}
```

3. Implementation details of "printArray" function: The function takes a pointer of type integer as input. Based on the value stored in the global variables "rowsize" and "colsize", an iterative loop should be constructed that would simply print all the values (tab separated in one line) stored in the array row wise. After printing all the elements in the row, a new line should be inserted so that the next subsequent rows are printed in different lines. The function does not return anything, and hence the return type should be void. I expect that you declare the function as follows:

```
void printArray(int *ptr) {
    ----
    ----
    Your logic goes here
    ----
}
```

4. Implementation details of "findMaxInRows" function: The function takes a pointer of type integer as input. First, the function should prompt the user to provide the row number. The row number starts with 1. Based on the value provided by the user for the row number, an iterative loop should be constructed that loop through all the elements in the corresponding row in the array and find the maximum value stored in the particular row of the given array. The function should return the maximum value of type integer. I expect that you declare the function as follows:

```
int findMaxInRows(int *ptr){
    ----
    ----
    Your logic goes here
    ----
}
```

5. Implementation details of "findMaxInCols" function: The function takes a pointer of type integer as input. First, the function should prompt the user to provide the column number. The column number starts with 1. Based on the value provided by the user for the column number, an iterative loop should be constructed that loop through all the elements in the corresponding column in the array and find the maximum value stored in the particular column of the given array. The function should return the maximum value of type integer. I expect that you declare the function as follows:

```
int findMaxInCols(int *ptr){
    ----
    ----
    Your logic goes here
    ----
}
```

6. Implementation details of "findMinInRows" function: The function takes a pointer of type integer as input. First, the function should prompt the user to provide the row number. The row number starts with 1. Based on the value provided by the user for the row number, an iterative loop should be constructed that loop through all the elements in the corresponding row in the array and find the minimum value stored in the particular row of the given array. The function should return the minimum value of type integer. I expect that you declare the function as follows:

```
int findMinInRows(int *ptr){
    ----
    ----
    Your logic goes here
    ----
}
```

7. Implementation details of "findMinInCols" function: The function takes a pointer of type integer as input. First, the function should prompt the user to provide the column number. The column number starts with 1. Based on the value provided by the user for the column number, an iterative loop should be constructed that loop through all the elements in the corresponding column in the array and find the minimum value stored in the particular column of the given array. The function should return the minimum value of type integer. I expect that you declare the function as follows:

```
int findMinInCols(int *ptr){
    ----
    ----
    Your logic goes here
    ----
}
```

A sample screenshot of the expected output is as follows:

```
ewire23-125:lab06 amohan$ ./twodim.out
Enter the rowsize:4
Enter the colssize:4
Enter the list of values in your row 1:
1      2      3      4
Enter the list of values in your row 2:
5      6      7      8
Enter the list of values in your row 3:
9      10     11     12
Enter the list of values in your row 4:
13     14     15     16
#printing the 2d array:
1      2      3      4
5      6      7      8
9      10     11     12
13     14     15     16
#Enter row number to find the maximum:3
#The maximum number is:12
#Enter column number to find the maximum:4
#The maximum number is:16
#Enter row number to find the minimum:2
#The minimum number is:5
#Enter column number to find the minimum:1
#The minimum number is:1
ewire23-125:lab06 amohan$
```

In order to get a maximum number in C, you may need to use the standard library called limits.h and the following code:

```
int maximum = INT_MAX;
```

The screenshot shown above is just for a sample. Your program should produce the correct result based on the user input values.

I expect that you would create a main function, that would orchestrate all the above 7 different function calls, to produce the output that matches the expected output.

Submission Details

1. Your lab assignment solution file must be uploaded to your repository by the due date and time.
2. Questions about the lab? Bring them to class on Tuesday morning!

Before you turn in this assignment, you must ensure that the course instructor has read access to your git repository that is named according to the convention `cs200F2018-<your user name>`.

Grading Rubric

1. If you complete Part 1 fully, as per the requirement outlined above, you will receive a total of 15 points. There will be 5 points awarded for each individual task in Part 1.
2. If you complete Part 2 fully, as per the requirement outlined above, you will receive a total of 35 points. There will be 5 points awarded for each individual task in Part 2.
3. Failure to upload the lab assignment code to your git repo, will lead you to receive "0" points given for the lab submission. In this case, there is no solid base to grade the work.
4. There will be a partial credit (30% of the points allocated for the part) given if your code in (Part 1, 2) compiles correctly but not able to produce the expected result.
5. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in github. In this way, an updated version of student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then in this case, there is 0 points to the student automatically for the lab work.
6. If you needed any clarification on your lab grade, talk to the instructor. The lab grade may be changed if deemed appropriate.

