CSCD 240

NOTE: Your answers, for all problems, will be saved in a file named cscd240Lab7.pdf for all problems

1) I have provided the following in cscd240Lab7.c and a Makefile. You must use my Makefile to compile your code.

```
int twod[4][3] = { {0, 1, 2}, {10, 11, 12}, {20, 21, 22}, {30, 31, 32} };
// NOTE: a 2D array is a double pointer so you have to derefernce the name
// to be able to use a 1D pointer
int *ptr = *twod;

/* This gives us an idea of the memory map */
printf("sizeof(ptr) %ld\n", sizeof(ptr) );
printf("sizeof(twod[0]) %ld\n", sizeof(twod[0]) );
printf("sizeof(twod[0][0]) %ld\n", sizeof(twod[0][0]) );

printf("twod %p\n", twod);
printf("ptr %p\n", ptr);

printf("&twod[0] %p\n", &twod[0]);
printf("&twod[0][0] %p\n", &twod[0][0]);
printf("&twod[0][1] %p\n", &twod[0][1]);
printf("&twod[0][2] %p\n", &twod[0][2]);

printf("&twod[1] %p\n", &twod[1]);
printf("&ptr %p\n", &ptr);
/* end memory map */
```

This code will provide a base address for twod and ptr.

Use the address of 0x7fffa46b3790 doe the starting location of the array
Use the address of 0x7fff24a09ed8 for the location of ptr.

Answer /complete the following

a) What is the size of ptr?
b) What is the size of twod?
c) What is the size of twod[0] and why?
d) What is the size of twod[0][0]?
e) What can you say about twod and twod[0] as it relates to the name of the array?
f) Draw a memory map that shows the memory locations of each element of the array and of ptr.

2) Using the provided address from #1 as the base address of the 2D array and the location of ptr, based on the code below, create an educated guess that clearly outlines what you believe will happen as each line is executed. In your explanation clearly explain what is happening, don't just give memory addresses or values. If you only provide memory addresses or values you will receive 0 points for this problem. Your guesses will be clearly labeled in the PDF file. You must provide the line of code and then the explanation. You must also provide per each line of code what the type is. Within each printf statements is a ?. You must specify if the ? would be a **d** for an int or a **p** for a pointer.

```
printf("twod + 3 is: %?\n", twod + 3);
printf("*(*(twod + 1)) is: %?\n", *(*(twod + 1)));
printf("*twod + 1 is: %?\n", *twod+1);
printf("*twod[2] is: %?\n", *twod[2]);
printf("*(twod + 2) + 2 is: %?\n", *(twod + 2) + 2);
printf("twod[1] is: %?\n", twod[1]);
printf("twod[1][2] is: %?\n", twod[1][2]);

printf("ptr %?\n", ptr);
printf("twod [1] %?\n", twod [1]);
printf("ptr[1] %?\n", ptr[1]);
printf("ptr + 1 %?\n", ptr + 1);
printf("*(ptr + 1) %?\n", *(ptr + 1) );
printf("twod + 1 %?\n", twod+1);
printf("*twod + 1 %?\n", *twod + 1);
printf("ptr[8] %?\n", ptr[8]);
```

3) Edit the C file

   a) Add the code from problem #2, make sure you replace the ? on each line with the appropriate symbol either a p or a d.

   b) Compile and execute – capture the output

   c) In the PDF clearly state the line of code, your guess and what the result was. If you guessed correctly then state – correct guess, otherwise clearly explain the incorrect guess. If you had the wrong symbol you must make a note explaining you had a d where there should have been a p or vice versa.

4) If the following line was added to the file printf("ptr[3][1] %d\n" do you think the code will compile? Why or Why not? You must specify an answer and you must justify your answer. If you don't justify you will receive 0 points for the part.

5) Add a function to your C file named function1. This function will take the 2D array as a parameter, and the array is passed using the square brackets. NOTE: you must specify the number of columns per row when passing the array using the square brackets. Using square bracket notation print out the values in the array. The print out will look similar to:

```
function1
  0  1  2
 10 11 12
 20 21 22
 30 31 32
```

   The function prototype: **void function1(int twod[][3], int rows, int cols);**

   The function call: **function1(twod, 4,3);**

6) Add a function to your C file named function2. This function will take the 2D array as a parameter, and the array is passed using the square brackets. NOTE: you must specify the number of columns per row when passing the array using the square brackets. Using pointer arithmetic notation print out the values in the array in reverse order for each row. The print out will look similar to:

```
function2
  2  1  0
 12 11 10
 22 21 20
 32 31 30
```

   The function prototype: **void function2(int twod[][3], int rows, int cols);**

   The function call: **function2(twod, 4,3);**

7) Add a function to your C file named function3. This function will take the 2D array as a parameter, and the array is passed as a 1D array using the square brackets. Using square bracket notation print out the values in the array. The print out will look similar to:

```
function3
  0  1  2
 10 11 12
 20 21 22
 30 31 32
```

The function prototype: **void function3(int twod[], int rows, int cols);**

The function call: **function3(twod[0], 4,3);**

8) Add a function to your C file named function4. This function will take the 2D array as a parameter, and the array is passed as a 1D array using pointer notation. Using pointer notation print out the values in the array in reverse order for each row. The print out will look similar to:

```
function4
  2  1  0
 12 11 10
 22 21 20
 32 31 30
```

The function prototype: **void function4(int twod[], int rows, int cols);**

The function call: **function4(twod[0], 4,3);**

9) Add a function to your C file named function5. This function will take the 2D array as a parameter, and the array is passed as a 1D array using pointer notation. Using square bracket notation print out the values in the array in reverse order for each row. The print out will look similar to:

```
function5
  2  1  0
 12 11 10
 22 21 20
 32 31 30
```

The function prototype: **void function5(int * twod, int rows, int cols);**

The function call: **function5(twod[0], 4,3);**

10) Add a function to your C file named function6. This function will take the 2D array as a parameter, and the array is passed as using both pointer notation and square bracket notation. Using square bracket notation print out the values in the array. The print out will look similar to:

```
function6
  0  1  2
 10 11 12
 20 21 22
 30 31 32
```

The function prototype: **void function6(int (*twod)[3], int rows, int cols);**

The function call: **function6(twod, 4,3);**

11) Explain the difference between **int (*twod)[3]** as a parameter as compared to **int *(twod[3])**. I am looking for a thoughtful explanation. Telling me something that is not thoughtful, such as the parentheses are different will earn you 0 points for this problem. Place this answer in your PDF.

12) Can we pass the array known as twod to a function such as the function call is **function7(twod, 4, 3);** where the prototype is **void function7(int ** twod, int rows, int cols);**? Why or why not? What happens if we try? Justify your answer. If you don't justify your answer then you will earn 0 points for this problem. Place this answer in your PDF.

13) In your PDF explain, the similarities and the differences of passing an array with the [] and passing the array as a pointer as it relates to a 1D array and a 2D array. HINT with a 2D array when you pass by [] you have to give the number of columns why? If you pass only by pointer how does that affect the use of the []? Explain you answer.

14) Compile and execute your code will functions 1 through 7 executing properly. Save your output as **cscd240Lab7out.txt**.

TO TURN IN:

A zip containing your Lab7 folder and :

- Your PDF
- Your C file
- Your output file
- My Makefile

You should know the naming scheme by now.