Jaime williams CSCD240lab9 due Thur 10/31

1.)

a) What is the size of ptr?

8 Bytes, this is an int** pointer explaining the size.

b) What is the size of twod?

48 Bytes, this addresses the memory for the array(see memory map).

c) What is the size of twod[0] and why?

12 Bytes 4 Bytes per col in the row.

d) What is the size of twod[0][0]?

4 Bytes

e) What can you say about twod and twod[0] as it relates to the name of the array?

Both properties are pointing at same address.

f) Draw a memory map that shows the memory locations of each element of the array and of ptr.

*** Memory Map ***
sizeof(ptr) 8
sizeof(twod[0]) 12
sizeof(twod[0][0]) 4
twod 0x7fffa46b3790
ptr 0x7fffa46b3790
&twod[0] 0x7fffa46b3790 //last col = int at address
&twod[0][0] 0x7fffa46b3790 0
&twod[0][1] 0x7fffa46b3794 1
&twod[0][2] 0x7fffa46b3798 2
&twod[1] 0x7fffa46b379c
&twod[1][0] 0x7fffa46b379c 10
&twod[1][1] 0x7fffa46b37a0 11
&twod[1][2] 0x7fffa46b37a4 12
&twod[2] 0x7fffa46b37a8
&twod[2][0] 0x7fffa46b37a8 20
&twod[2][1] 0x7fffa46b37ac 21
&twod[2][2] 0x7fffa46b37b0 22
&twod[3] 0x7fffa46b37b4
&twod[3][0] 0x7fffa46b37b4 30
&twod[3][1] 0x7fffa46b37bb 31
&twod[3][2] 0x7fffa46b37bc 32
&ptr 0x7fffa46b3788

2.)

printf("twod + 3 is: %p\n", twod + 3);

type = int** val = 37b4 this adds 24 bytes (3 ptr increments) to 3790, which is address of twod[3].

printf("*(*(twod + 1)) is: %d\n", *(*(twod + 1)));

type = int, val = 10. Derefernces int** gets int at that location (0) and moves the pointer by 1 size 8 Bytes which returns value at twod[1] address (same as twod[0][1]).

printf("*twod + 1 is: %p\n", *twod+1);

type = int*, val = 3794. *twod is still an int*. This pointer is looking at the row of cols and repoints from twod[0][0] to twod[0][1].

printf("*twod[2] is: %d\n", *twod[2]);

type = int, val = 20. Dereferences twod and displays int at twod[2][0].

printf("*(twod + 2) + 2 is: %p\n", *(twod + 2) + 2);

type = int*, val = 37b0. redirects to twod[2] then redirects to twod[2][2].

printf("twod[1] is: %p\n", twod[1]);

type = int*, val = 379c. returns address for twod + 1 (double pointer only dereferenced once with [] math).

printf("twod[1][2] is: %d\n", twod[1][2]);

type = int, val = 12. returns int value for twod[1][2].

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

type = int*, val = 3790. ptr base address.

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

type = int*, val = 379c. Add 8 Bytes to twod[0] address.

printf("ptr[1] %d\n", ptr[1]);

type = int, val = 1. ptr is pointing at int in twod[0][1].

printf("ptr + 1 %p\n", ptr + 1);

type = int*, val = 3794. Increments pointer 1 unit sizeof(twod[0][0]) which is 4 Bytes.

printf("*(ptr + 1) %p\n", *(ptr + 1));

type = int*, val = 3794. Adds 4 Bytes to twod[0][0] address resulting in retuning adress of twod[0][1].

printf("twod + 1 %p\n", twod+1);

type = int**, val = 379c.

printf("*twod + 1 %p\n", *twod + 1);

type = int*, val = 379c. Pointer increments to twod[1].

printf("ptr[8] %d\n", ptr[8]);

type = int, val = 22. moves pointer to return val at twod[2][2].

3.)

1st guess:

```
printf("twod + 3 is: %p\n", twod + 3);
printf("*(*(twod + 1)) is: %d\n", *(*(twod + 1)));
printf("*twod + 1 is: %p\n", *twod+1);
printf("*twod[2] is: %d\n", *twod[2]);
printf("*(twod + 2) + 2 is: %p\n", *(twod + 2) + 2);
//error printf("twod[1] is: %d\n", twod[1]);
        incorrectly assumed [] would resolve to an int.
printf("twod[1][2] is: %d\n", twod[1][2]);
printf("ptr %p\n", ptr);
printf("twod [1] %p\n", twod [1]);
// error printf("ptr[1] %p\n", ptr[1]);
        incorrectly assumed ptr[1] was an int*.
printf("ptr + 1 %p\n", ptr + 1);
//error printf("*(ptr + 1) %p\n", *(ptr + 1) );
        incorrectly assumed *(ptr + 1) was a double pointer.
printf("twod + 1 %p\n", twod+1);
printf("*twod + 1 %p\n", *twod + 1);
//error printf("ptr[8] %p\n", ptr[8]);
        using the [] to pull int not int*.
```

working code:

```
printf("twod + 3 is: %p\n", twod + 3);
printf("*(*(twod + 1)) is: %d\n", *(*(twod + 1)));
printf("*twod + 1 is: %p\n", *twod+1);
printf("*twod[2] is: %d\n", *twod[2]);
printf("*(twod + 2) + 2 is: %p\n", *(twod + 2) + 2);
printf("twod[1] is: %p\n", twod[1]);
printf("twod[1][2] is: %d\n", twod[1][2]);
printf("ptr %p\n", ptr);
printf("twod [1] %p\n", twod [1]);
printf("ptr[1] %d\n", ptr[1]);
printf("ptr + 1 %p\n", ptr + 1);
printf("*(ptr + 1) %d\n", *(ptr + 1) );
printf("twod + 1 %p\n", twod+1);
printf("*twod + 1 %p\n", *twod + 1);
printf("ptr[8] %d\n", ptr[8]);
```

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.

This will not work because ptr is not a double pointer

5.) see cscd240Lab9.c
6.) see cscd240Lab9.c
7.) see cscd240Lab9.c
8.) see cscd240Lab9.c
9.)Answer to NOTE: twod is actually a double pointer
10.) see cscd240Lab9.c

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.

It appears that the order of operations is in play. (*twod)[3] dereferences twod int** Then uses [3] to return int at twod[0][3], *(twod[3]) dereferences the result of twod[3] returning twod[3][0].

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.

No we cannot. The compiler does not view the int** as a compatible pointer type. It compiles but instead of passing twod pointer it passes ints.

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.

C works with all arrays as if they were a single array, by adding the columns to the pointer; the program understands where each individual rows address is located in memory.