TPS (Think-Pair-Share) activity 1: Discuss questions 1 – 10 (25 minutes) while paired with your classmates in a breakout room assigned by your TA (you will be assigned to groups of 3-4 students) and record your answers in a text file named tpsAnswers.txt under a section labeled "TPS 1" (you will continue to use this file to record your answers to all the TPS questions that follow in the lab handout):

1. Open MemCast.c, compile, and run the program. What do you expect the program to print? (%x in printf allows an integer to be printed in Hex format).

- %x will print integer four_ints[0]=2 in hex format i.e. 2

2. Before changing the code, what do you expect the program to print if you print four_ints[0] again at the end of the program?

- It will print four_c[i] = 'A'; // ASCII value of 'A' is 65 or 0x41 in Hex. four_c -> 41 41 41 41 i.e. 4 bytes so this will change array four_ints first location i.e. four_ints[0] = 41414141 as int is 4 bytes long. Its this due to the loop.

3. Insert a print statement to output four_ints[0] at the end of the program and verify your answer from (2).

- printf("%x\n", four_ints[0]);

4. Now add a print statement to the program so it will print out four_ints[1]. What does it print? Are you surprised (or lost) by the results?

- 41414141

5. Let us study the code carefully and investigate what happened. No, the memory did not go crazy.

a. How many array(s) were allocated in this program?

- Only one array, which is four_ints[4]

b. Are four_ints and four_c pointing to the same location?

- Yes. the array name is pointing to the base address of the array. So both four_c and four_ints points to the same location i.e. the base address of the array.(four_c = (char*)four_ints;)

c. Verify your answer of (b) by printing out the values of four_ints and four_c.

- Yes they both show the same address.

6. Write a loop to print out the addresses and values (in Hex) of all the elements of four_ints. What is the difference in addresses between two consecutive elements? Discuss what this difference means.

- The difference between the two is that one is just the location where the data being stored and the other one is the hex version of the value stored within the memory block.

7. Use a piece of paper to draw the layout of the array horizontally so that the smallest address begins from the RIGHT-hand-side. Indicate the address and value of each element based on the results of (6). You can draw it digitally.

four_ints[i]

3	2	1	0
value: efbff4dc	value: efbff4d8	value: efbff4d4	value: efbff4d0
address: 0x7ffeefbff4dc	address: 0x7ffeefbff4d8	address: 0x7ffeefbff4d4	address: 0x7ffeefbff4d0

8. Now, write a loop to print out the same information for four_c as you did in (6). What is the difference in addresses between two consecutive elements? Discuss what this difference means.

- The reason that is the same as in number 6, only thing is that with this one, we are dealing with char types.

9. Use the same piece of paper (or file) from (7) to draw a similar structure for four_c.

four_c[i]

3	2	1	0
value: efbff4d3	value: efbff4d2	value: efbff4d1	value: efbff4d0
address: 0x7ffeefbff4d3	address: 0x7ffeefbff4d2	address: 0x7ffeefbff4d1	address: 0x7ffeefbff4d0

10. By comparing the layout of the array pointed by the two pointers, what do you observe in terms of how C accesses memory when the index of an array is incremented?

- I can observe that depending on the data type of the array or the array pointer, it will either increment by either 4 bytes, or by only 1 byte, whether it be an int type, or a char type

TPS activity 2: Discuss questions 1 – 7 (25 minutes) with your TPS partners in your assigned breakout room and record your answers in tpsAnswers.txt under a section labelled "TPS 2":

1. Open Array2D.c. This program will create a n x n array of int. Explain what line #8 does.

- When opening the file, Array2D.c, you can see that line 8 is used to set a double int pointer as an array, to implement a library function, and set the sizeof. Also, Line number 8 is allocating enough memory for a pointer for the double pointer **arr of type int.

2. Since every array must be allocated in the memory before it can be used, we need to allocate the rows one by one. To do this, we need to be able to access the pointers from the first array (pointed by arr). Assuming i is the index of that array, how do we access the ith value of the array?

- In order to access the ith value of the array you need to i = 0 and access the pointer that is pointing poiting to the that value of the array. Also, To access the ith value of the array we just have to type the following code: "*(arr + I) = (int *) malloc(n * sizeof(int));

3. Without using array notations ([]), insert code to complete allocating all the rows and initialize all contents to be 0. Your code should work with different values for n. Hint: if j is the index of each row, how do you access arr[i][j] in pointer notation?

- **In order** to access the array, arr[i][j] in pointer notation you need to include your print Arr which is two for loops, and to access the arr[i][j] you do int i,j. Also, we do assume there is a for loop, so we have to include this code: "*(*(arr+i)+j) = some value;"

4. To verify whether you have created your array correctly, we need a function to print out the array. The function printArray has been declared. It takes in both the array to be printed and the size of the array. Why do we need to pass the size as an argument?

- We have to pass size as an argument because there needs to be a change of information. Argument refers to any expression within the parentheses of a function call.

5. Complete printArray so it prints out the content and layout of an array correctly.

6. Now, let us modify the content of the array. Insert code to make the array into a diagonal matrix that looks like the following (again, do not limit the size to 5):

1	**0**	**0**	**0**	**0**
0	**2**	**0**	**0**	**0**
0	**0**	**3**	**0**	**0**
0	**0**	**0**	**4**	**0**
0	**0**	**0**	**0**	**5**

7. Call printArray to print out your new array and verify the result.