Andre Martin

Partner: Omar Silva

**Lab 4 – Memory in C**

**TPS (Think-Pair-Share) Activity 1:**

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

1. 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?

* We are casting int pointer to char pointer i.e. four\_c = (**char**\*)four\_ints; four\_c is character pointer and will write byte by byte.

**For** (i=0; i<4; i++)

four\_c[i] = 'A'; // ASCII value of 'A' is 65 or 0x41 in Hex.

after this four\_c will point to array with contents

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.

so if we print four\_ints[0] again then it will print 41414141

1. Insert a print statement to print out four\_ints[0] at the end of the program and verify your answer from (2).

* The output of program is 2 41414141   <---- printf at the end

1. 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?

* Printf("%x\n", four\_ints[1]); this will print 2. Since we changed only four\_ints[0] to 41414141 using char pointer.

5. Let’s 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 1 array is allocated i.e. four\_ints[4]

b. Are four\_ints and four\_c pointing to the same location?

- Yes. array name is pointing to base address of array. So, both four\_c and four\_ints points to same location i.e. base address of array. (four\_c = (char\*) four\_ints;)

c. Verify your answer of (b) by printing out the values of four\_ints and four\_c.

- Done

1. Write a loop to print out the addresses (int Dec) 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 of where the data is being stored and the other one is the hex version of the value stored within the memory block.

1. 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.

- Done

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

- The reason 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 of four\_c.

- Done

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 (Think-Pair-Share) Activity 2:**

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

- When you open Array2D.c line 8 is used to set a double inter 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?

- To access the ith value of the array you need to i = 0 and access the pointer poiting to the that value of the array. Also, To access the ith value of the array we just have to type this, “\*(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. You 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?

- To access arr[i][j] in pointer notation you need to implement your print Arr which is two for loops, and to access the arr[i][j] you do int i,j. Assuming there is a for loop, we have to include this, “\*(\*(arr+i)+j) = some value;”

8.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 size of array. Why do we need to pass the size as an argument?

-We need 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. Done

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 result. Done

**(Assignment 1, individual) Matrix multiplication:**

1. Open MatrixMult.c. and define in main () two n x n matrices (arrays) using malloc. Done
2. Implement printArray function so you can call it to print a 2D array. Done
3. In main (), call printArray to print out the 2 arrays you have defined in (1). Done
4. Implement matMult so it multiplies the 2 input arrays and return the resulting array. Pay attention to the input arguments and return type. Done
5. In main (), call matMult to multiply the 2 arrays you have defined in (1). Done
6. In main (), call printArray to print out the resulting array you receive in (5). Done
7. You need to declare any variables that are necessary. Done