Day 24 Highlights

1. Reminders
   1. Project 4 due this Friday at 5pm
2. One-dimensional arrays
   1. Can declare arrays either on the stack or on the heap

**int array1[100];**

**int \*array2;**

**array2 = malloc(sizeof(int) \* 100);**

* 1. When defining functions, can use [ ] or \* notation

**int funct1(int \*array, int size) {}**

**int funct1(int array[], int size) {}**

**in main: x = funct1(array, 100);**

* 1. Use **free** to free the memory allocated by malloc

1. Two-dimensional arrays on the stack
   1. Declaring and Using two-dimensional arrays in the same function is no big deal

**int y[2][3] = { { 11, 12, 13}, {21, 22, 23} };**

**int y[2][3];**

**y[1][1] = 37;**

**y[0][1] = y[1][0];**

* 1. Passing two-dimensional arrays to functions is problematic  
     Passing 2D arrays works – but the number of columns has to be known

**int funct1(int array[][10], int rows) {}**

* 1. Explain how the system needs to know the dimensions in order to figure out the position from the subscripts (2-d arrays are stored as 1-d arrays in row-major order. For example, **array[r][c]** is stored at location **10\*r+c** in a 1-d array.)
  2. Show how [1][2] and [1][0] are the same thing in the program below (make them realize how arrays are stored)

**int getElement(int x[][6], int r, int c)**

**{**

**return x[r][c];**

**}**

**int main(void) {**

**int array[6][4] = { { 1,2,3,4}, {5,6,7,8},**

**{9,10,11,12}, {13,14,15,16},**

**{17,18,19,20}, {21,22,23,24} };**

**while (1) {**

**printf("Enter two subscripts or CTRL-D to exit : ");**

**int a, b;**

**scanf("%d%d", &a, &b);**

**if (feof(stdin)) break;**

**printf("array[%d][%d] is %d\n", a, b, array[a][b]);**

**int pos=4\*a+b;**

**int r=pos/6;**

**int c=pos%6;**

**printf("array[%d][%d] is %d\n", r, c,**

**getElement(array, r, c));**

**}**

**return 0;**

**}**

1. Solution: view a 2D array as an array of 1D arrays and allocate the space on the heap
   1. Declare the array

**int \*\*array;**

* 1. Allocate the “backbone” (one item for each row)

**array = malloc(sizeof(int \*) \* numRows);**

* 1. Allocate “all the columns for that row”

**for (a=0; a<numRows; a++)**

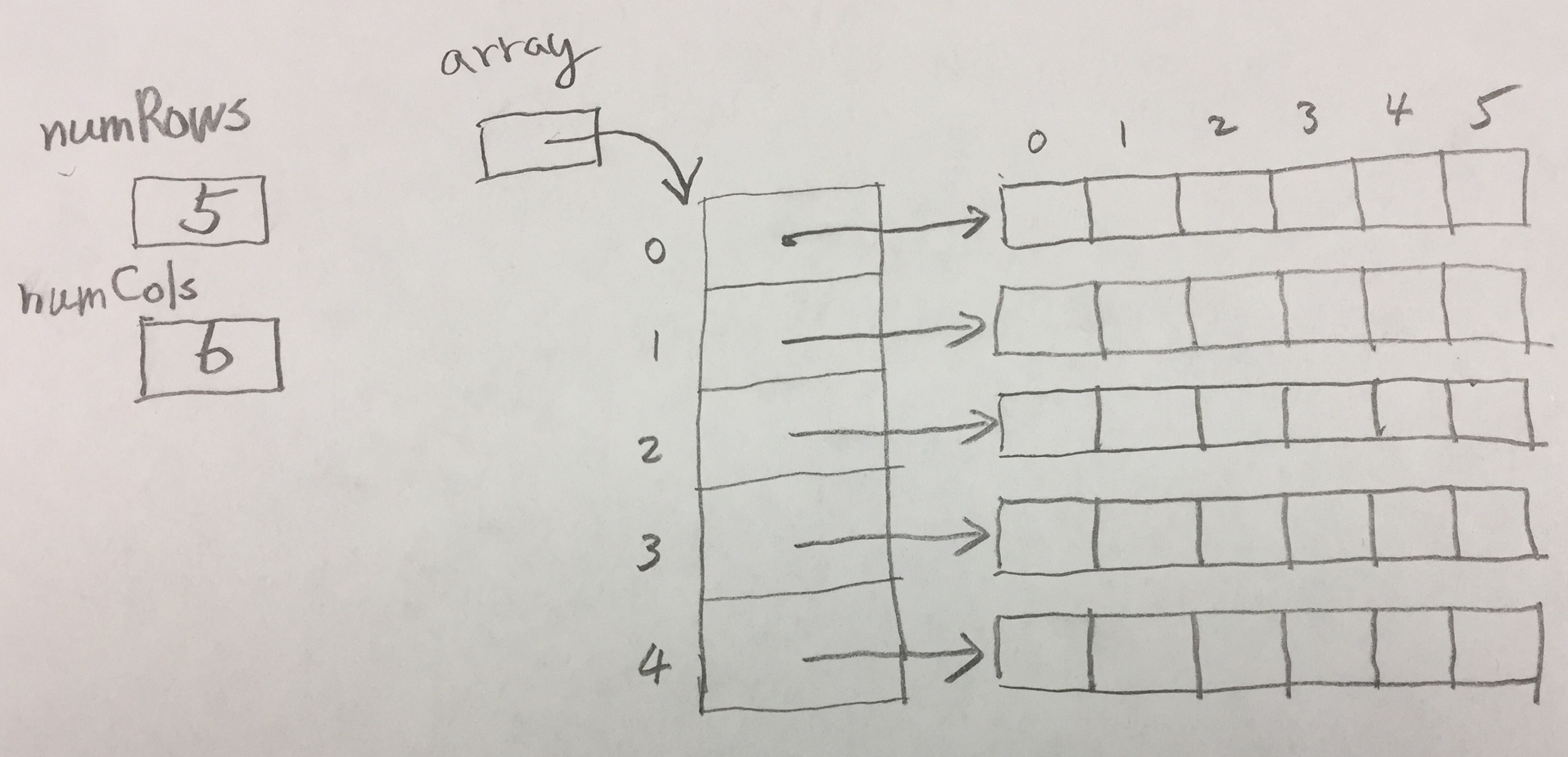
**array[a] = malloc(sizeof(int) \* numCols);**

1. Write functions to allocate space for a 2D array, to populate a 2D array using a random number generator, to print an array.

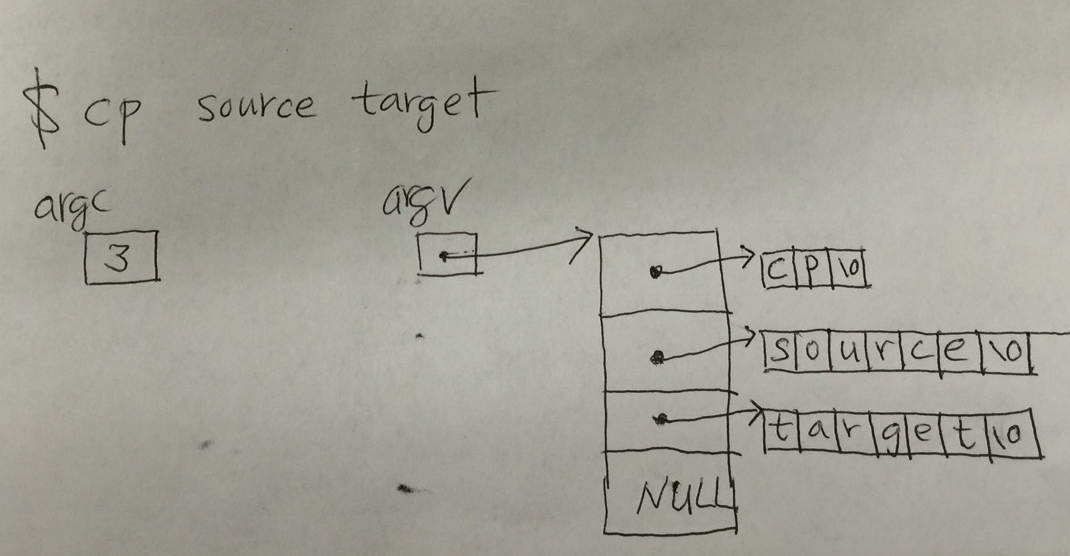
**Solution: A 2D array is an array of 1D arrays.**

int numRows, numCols;

int \*\*array;

****

int main(int argc, char \*\*argv) {}



What is argv[2][4]?