LAB #7 – Pointers and Arrays

Understanding Pointers:
As part of this lab, you are to pair with someone else in the lab, and run tests on pointers to make sure you understand what is happening in memory. Create a pointer to a double, i.e. `double *d;`, and three doubles `d1, d2, and d3`, i.e. `double d1, d2, d3;`, that get the values 7.8, 10.0, and .009.

Now, set the pointer, `d`, to point to each double variable, `d1, d2, and d3`, and print the address of each variable, along with the contents of the double variable using `d`. What if you made a pointer that points to a double pointer, i.e. `double **dp`? Now, set the pointer to a double pointer, `dp`, to point to `d`, the double pointer, and use `dp` to print the address and contents of each double variable!!!

Practice Arrays/Command Line Arguments:
In this lab, you can practice with taking in command line arguments for the `n` and `r` values in your previous lab. **You can decide to supply them in a specific order, or more realistically, supply them in any order. Use the options –r and –n to denote when the n and r are being entered, i.e. `./combinations –n 100 –r 3`.

In order to read command line arguments, you need to alter your `int main()` function. You need to add `int main(int argc, char *argv[])`. The `argc` parameter is the count of arguments passed to the program, including the program name, i.e. `argc is 5`, and you can access the name of the program and arguments provided to the program using `argv`, which contains the values of the arguments including the name of the program, i.e. `argv[0] points to ./combinations`. The `argv` parameter is a 2-d array, i.e. `argv[1][0] and argv[1][1] refer to the – and n characters`, so you will need to convert the characters to a number to use with factorial. Maybe now is a good time to check that the numbers supplied where integers before converting them.

Now, try reading the `n` and `r` values using command line arguments, where –n and –r can come in any order, i.e. `./combinations –r 3 –n 100`, but the two have to be together, i.e. option then number. If the user enters these options incorrectly or invalid options, you should print an error message indicating how to use the program. For example:

`./combinations –n –r 100 3`

Usage: `./combinations [–n <number>] [–r <number>]`

Notice that there are square brackets around the options in the usage above, this means that these are optional options. If the user omits one option or both options, then your program should prompt the user for these missing pieces of information.

Show your two programs to a lab TA for full lab credit.
Extended Learning:

Change your Assignment #4, so that the user enters a sentence and option to the program as command line arguments. You'll find out whether the user wants to determine if the string is a palindrome or to reverse the string by taking the option –r or –p, and if the user doesn't enter an option, then he/she will be prompted to reverse or determine if it is a palindrome.

Remember, if you want to enter a sentence, then you must enter the surround the sentence in double quotes, i.e. “hello world”.