ECE 71 – Engineering Computations in C

Professor Kriehn – Fall 2016

Code Due By: Midnight on Tuesday, October 18, 2016
Writeup Due By: Class on Thursday, October 20, 2016

HOMEWORK #19 – Phone Number Translator

Write a program that translates an alphabetic phone number into numeric form:

```
Enter Phone Number:     CALLATT
Numeric Form:           2255288
```

In case you do not have your cell phone handy, remember that: 1 = .,@, 2 = ABC, 3 = DEF, 4 = GHI, 5 = JKL, 6 = MNO, 7 = PQRS, 8 = TUV, 9 = WXYZ. If the original phone number contains non-alphabetic characters besides the ones listed for the number 1 ('.', ',', or '@'), your program should leave the results unchanged. For example:

```
Enter Phone Number:     1-800-COL-LECT
Numeric Form:           1-800-265-5328
```

You may assume that any letters entered into your program by the user could upper or lower case.

Specifications:
The program needs to store the original phone number in array of characters. The program also needs to store the numeric number in a second array of characters. You may assume that the phone number is no more than 15 characters long, and any letters that are entered into the program by the user may be upper or lower case. Define a constant N for your array size and use it throughout your program.

If you execute the program, the following information should be displayed:

```
~> hw19.o
Enter Phone Number:     1-800-765-5234
Numeric Form:           1-800-765-5328
~> hw19.o
Enter Phone Number:     1800KrIeHn7
Numeric Form:           18005743467
~> hw19.o
Enter Phone Number:     011886INTERNTL.
Numeric Form:           011886468376851
~>
```

HINT: You will need to initialize your two arrays with the NULL character (ASCII Code 0) so that if you do not print out 15 numbers, you will not print out junk values left over in other elements in the arrays. You will need to use a `while()` loop to read in your characters, and use a `for()` loop for your conversion. You may also find a `switch()` statement to be extremely useful…

HOMEWORK #20 – 2-D Array Generation

Write a program that creates (and stores) a 2-D array of 25 random integers (5 rows and 5 columns) between 0 and 100. The program should then print out elements in the array, as well as the element number (ROW and COLUMN position) of the largest item in the array (the first time it occurs if there is more than one maximum) as well as its value.

Specifications:
Use constants to define the number of rows and columns within your array. For this program, do not use the time trick we learned about in class to seed the random number generator. For the sake of the grader program, prompt the user to enter a number for the random seed, store it as an unsigned integer, and use it to seed your srand() function. See the example code on my website to help you with this if you get stuck.

If you execute the program, the following information should be displayed:

```
~> hw20.o
Please enter a random seed: 1000
5 x 5 Random Array
  26  78  47  86  98
  30  72  57  20   7
  22  65  73  61   8
  27   5  22  23  26
  97  46  55  93   6

The maximum value occurs at element: 0, 4
The maximum value is: 98
~> hw20.o
Please enter a random seed: 312
5 x 5 Random Array
  64  52  27  83  16
   6  94  10  89  27
  56  43  25  29   4
   3  93  17  28  94
   3  36  59   3  25

The maximum value occurs at element: 1, 1
The maximum value is: 94
```

Notice that the second tie the program is run that there are two 94s, but the first one occurs at element position ROW 1, COLUMN 1.

Please also note that the random numbers generated using the seed values on my laptop are going to be different than the random numbers generated using the seed values on the server. Use this example output as a guide for your program, but do not expect the random numbers that are being generated by your program to be the same if you use the same seed values used in these two examples.

Please do not wait until Monday to start this homework…