**Assignment #2 (Extend-O-Credit eligible)**

**Objectives:**

* Creation and use of functions
* Passing arguments into parameters
* Splitting your code into modular functions
* Making appropriate design decisions
* Following stated requirements

**Overview of Requirements**

* Write a program that uses a function to calculate the result of raising one number to the power of another.
* Use an on-screen menu to prompt for user input.
  + The menu must look like this:

Power Menu:

1. Change base
2. Change exponent
3. Display base raised to exponent
4. Exit program

Option?

For example, the function should be able to compute the fact that 10 to the power 3 is 1000 and that 20 to the power 4 is 160000. Test your program with more than the above two sets of values.

**Detailed Functional Requirements**

**Actions:**

* The user must be prompted for the base or exponent whenever they choose that option. The result must be calculated and displayed on user request through a menu option (as above).
* The program **must loop and must only exit** when the appropriate menu item is chosen.

**Calculation:**

The power calculation must be done using a function that you create. Requirements for this function:

* It must be called calculatePower.
* It must take two parameters (the base and exponent).
* It must return the calculated value to the calling function.
* It must use a loop to calculate the power. Multiple if-else statements are not permitted in this function!

**Values:**

* Initial values for the base and exponent must be 1.
* The base and exponent values are always integers.
* The base number (that is, the number which is being raised to the exponent) must only be in the range 1 to 25.
* The exponent must only be in the range 1 to 5.
* You can assume that the user will not be hostile and won't try to enter a number that is larger than the largest possible int.
* Range checking for the base and exponent must be done using a function that you create. This is one function that is called twice, once for checking the base, once for checking the exponent.  Requirements for this function:
  + It must be called checkRange.
  + It must take at least three parameters (value to check, minimum acceptable value, maximum acceptable value). You can have more parameters if you wish (but you really should be very sure about your design if you add more).
    - This implies that you should not create one range checking function for the base and a second one for the exponent **unless** all those functions do is have a single line that calls checkRange with the appropriate arguments.
  + It must return 0 if the value is out of range and 1 if the value is within the specified range.
  + It must **not** display any output. The calling function must print an error message if the value is out of range.
* Out-of-range bases and exponents must restore the previous value.
  + Do **not** reprompt the user for a valid base or exponent if they enter one that is out of range.
  + As stated in an earlier requirement, display an error message if the value is out of range (but not in calculatePower).

**Other Requirements**

* It must not use scanf() or fscanf(). Using cin is discouraged unless you handle the issues associated with using cin (which you should be aware of already if you're thinking of using cin).
* It must not use global variables (covered in the Scope and Style lecture).
* It must not use goto.
* It must not use the pow() function.
* You must not call the main() function recursively (in other words, don't have a call to main() within any of your functions).
* Appropriate programming style as discussed in class and in the Course Notes must be used.
* Function prototypes for all functions you create (except main()) must be used.

**File Requirements:**

* Call your project cA2.
* You must call your source file cA2.c or cA2.cpp to generate cA2.exe.
* Call your ZIP file cA2.zip.

**getNum():**

You can (and should) use a function such as the following to get input:

#pragma warning(disable: 4996)

int getNum(void)

{

/\* the array is 121 bytes in size; we'll see in a later lecture how we can improve this code \*/

char record[121] = {0}; /\* record stores the string \*/

int number = 0;

/\* NOTE to student: indent and brace this function consistent with your others \*/

/\* use fgets() to get a string from the keyboard \*/

fgets(record, 121, stdin);

/\* extract the number from the string; sscanf() returns a number

\* corresponding with the number of items it found in the string \*/

if( sscanf(record, "%d", &number) != 1 )

{

/\* if the user did not enter a number recognizable by

\* the system, set number to -1 \*/

number = -1;

}

return number;

}

This function should be put at the end of your source file.

Make sure that you create a function comment and prototype for getNum(). Reformat the bracing and indentation of getNum if required to match your own style.