SENG1000 – C/C++ PROGRAMMING

MAJOR ASSIGNMENT 2 -CREATING AND USING FUNCTIONS

OVERVIEW

Write a program that calculates powers using a function that you create.

GENERAL COURSE OBJECTIVES ADDRESSED IN THIS ASSIGNMENT

- Use arithmetic operators.
- Use logical operators.
- Use if statements.
- Use a loop.
- Use pre-existing functions.
- Create functions.
- Pass arguments into parameters.
- Split your code into modular functions.
- Make appropriate design decisions.
- Follow stated requirements.

ACADEMIC INTEGRITY AND LATE PENALTIES

- Link to Academic Integrity Information
- Link to Late Policy

EVALUATION

- The evaluation of this assignment will be done as detailed in the Marking lecture from Week 2.

PREPARATION

- View Week 3 videos.

REQUIREMENTS

MENU

- Use an on-screen menu to prompt the user for input.
- The menu **must** look like this (spacing can vary but the wording **must** be identical):

```
Power Menu:
   1. Change base
   2. Change exponent
   3. Display base raised to exponent
   4. Exit program
Option?
```

  - The rationale for this is that, in industry, you will often be given exact specifications by UI designers and you would be expected to follow them precisely.
- The program must loop, displaying the menu each time that menu input is required. Only quit the program when the user selects menu item 4.
  - Do not use the exit() function.

- There must be two blank lines prior to the "Power Menu:" line to separate it from the previous output.
- If the user enters an invalid menu entry, display "Invalid entry" and continue with the menu loop.

CALCULATION:

- Calculation is performed **only** when the user selects menu item 3.
- The power calculation must be done using a function that you create. Requirements for this function:
    - It must be called powerCalculation.
    - It must take exactly two parameters (the base and exponent, in that order, as ints).
    - 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!
        - The reason for disallowing multiple if-else statements is that it would no longer be practical if the maximum exponent was something higher, such as 100. This is a principle called "scalability".
        - This also implies that you cannot use the pow() function. Using the pow() function will cap your mark at 40 with resubmissions not being allowed.
    - Do not display anything in this function.
        - The rationale for this is that displaying output in a calculation function would limit its reusability in other programs.
    - It is important that you meet the requirements when provided for defining functions exactly, as (in industry) they often are provided as definitions as part of a larger system. Deviating from the requirements can break the system.
- Display the result of the calculation in main(). The wording is up to you.

VALUE CONSTRAINTS:

- The base and exponent variables must be int variables declared in main(). They must have initial values of 1.
- 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. They will also not enter values that are not 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.
- When getting the base or exponent from the user, make sure to prompt them appropriately.
- 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 rangeChecker.
    - It must take exactly three parameters (value to check, minimum acceptable value, maximum acceptable value).

    - 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 rangeChecker 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.
- Out-of-range bases and exponents **must** restore the previous value as appropriate.
    - Do **not** reprompt the user for a valid base or exponent if they enter one that is out of range.
- In main(), display an error message if the value is out of range (**not** in powerCalculation or rangeChecker).

INPUT USING GETNUM():

- You must use the following function to get input:

```
#pragma warning(disable: 4996)
int getNum(void)
{
/* we will 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.
    - Failing to put this function definition in your source file will result in your program not building properly.
- Make sure that you create a function comment and prototype for getNum(). Reformat the bracing and indentation of getNum to match your own style.

COMMENTING AND INDENTATION:

- You must have a header comment similar to that found in the SET Coding Standards or the Course Notes. This requirement is the same as in Focused Assignment 2.
- You must have function comments for all functions except main(). This includes getNum().
- Indent bodies as previously discussed in lecture.

OTHER REQUIREMENTS:

- Use the SET Coding Standards (found on eConestoga) that are relevant.
- The program must not use scanf(), fscanf(), scanf_s(), or fscanf_s().
- The program must not use goto (this requirement holds for all subsequent assignments).
- All variables must be declared within functions (i.e. it must not use global variables (covered later in the Scope and Style lecture)).
- Appropriate programming style as discussed in lecture and in the Course Notes must be used.
- **Do not have any input or output except as required by this assignment.**
- It is assumed that you will adhere to all course requirements detailed in the Course Notes readings so far in the course. **This requirement holds for all subsequent assignments.**
- Do not clear the screen (this requirement is true for all assignments).

CHECKLIST REQUIREMENTS

- Create a requirements checklist. This should contain the specific requirements from this assignment as well as any relevant requirements that have been covered in lecture or that are found in the SET Coding Standards or SET Submission Standards. Do it in whatever form you wish. Hand in your completed checklist in PDF form as checklist.pdf. Not having this checklist will result in a cap of 80 on your mark.

FILE NAMING REQUIREMENTS

- You must call your source file m2.cpp.
- You must call your checklist checklist.pdf.

SUBMISSION REQUIREMENTS

- Do not hand in any other source files besides those mentioned in the File Naming Requirements.
- Follow the instructions in the SET Submission Standards and the lecture on Submitting Assignments to submit your program. Submit both files in one submission to the correct Assignment folder.
- Once you have submitted your files, make sure that you've received the eConestoga e-mail confirming your submission. Do not submit that e-mail (simply keep it for your own records until you get your mark).

ADDITIONAL INFORMATION

- As an illustration, the powerCalculation() 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.
- The rangeChecker() function can be used more than twice or in other assignments if you wish. This is an excellent function to show the usefulness of reusability.

- In assignments in general, if you are required to display an error message but the text of the message is not specified in the requirements, you can choose what that wording is. Be reasonable in your choice and do not insult the user (you'd think that this would be unnecessary to say but it's not).