# Project Development Paper

**By: Steven Davis**

**Class: CSC 414 Software Design**

**Due Date: November 17, 2020**

# Project Description

## 1.1 Identifications

This program is called “Multitool Menu” and is version 1.0.

## 1.2 System Overview

The purpose of this program is to provide a simple suite of functions that may be useful to the end user, while also being easily modifiable should the user need to add new or remove old functions.

## 1.3 Document Overview

This document is going to contain the requirements, design process, and testing results of the simple program.

## References

NONE

# System and other requirements

| Requirement Identifier | Requirement Statement | Testing Identifier | Pass/Fail Testing |
|---|---|---|---|
| **1.0** | **Factorial** | | |
| 1.1 | Able to calculate factorials | 1a, 1b, 1c | P |
| **2.0** | **Addition** | | |
| 2.1 | Able to add together two numbers | 2a, 2c | P |
| 2.2 | Able to add together as many numbers as the user wants | 2b, 2d | P |
| **3.0** | **Multiplication** | | |
| 3.1 | Able to multiply together two numbers | 3a, 3c | P |
| 3.2 | Able to multiply together as many numbers as the user wants | 3b, 3d | P |
| **4.0** | **Powers** | | |
| 4.1 | Able to take a number and raise it to a power | 4a, 4b, 4c | P |
| **5.0** | **Year Check** | | |
| 5.1 | Able to determine if an inputted year is a leap year | 5a, 5c | P |
| 5.2 | Able to determine if an inputted year is not a leap year | 5b, 5d | P |
| **6.0** | **Random Number Generation** | | |
| 6.1 | Able to generate a number from 1 to 6 | 6a | P |
| 6.2 | Able to generate a number from 1 to 20 | 6b | P |
| 6.3 | Able to generate a number from 0 to 10,000 | 6c | P |

# Program Design

For the sake of modularizations in this project, I chose to give many part of the project their own functions. This way, should the project need to be modified in the future, the programmer would only need to change code within that specific block, rather than changing something that could affect code further down the line.

For the "main" block in the program, I simply codded in the function call to a menu functions, which itself calls a function to output the menu. After that function finishes, the main menu functions will ask for an input, which is then filtered through a switch statement. Each case of the switch will call the appropriate function to run the request, then return back to the menu function, which loops until the exit condition is selected.

For the Factorial calculator, I declare an input variable, which acts as a loop control, and an accumulator variable that is declared as 1 at the start. After the input is given, a For loop begins, in which the counter variable for the For loop. The current value of the accumulator is multiplied by the counter, and the result is stored in the accumulator. This continues until the counter exceeds the value inputted. Once the loop is done, a simple output is run.

The two addition functions are rather simple. The functions to add together two number simply takes two inputs, adds them together, and runs an output. For the function that takes in as many numbers as requested, it takes an input variable to act as the loop control, limiting how many times the For loop is run. I also declare an accumulator outside the loop and initialize it to 0. Within the loop, an input is requested, then added to the accumulator. Once the loop finishes, the output is run. The two multiplication functions act much the same, with the difference of using multiplication instead of addition.

For the power calculator, I took advantage of the "cmath" library. The function itself requests the inputs of the power and the base number. It then feeds those numbers into the "pow" function of the "cmath" library. The function does the calculation with me needed to waste space coding in a loop to do the calculation.

The leap year calculator function takes an inputted year, and runs I through an if-else "net". If the inputted year is divisible by 400, the functions says it's a leap year. If the year isn't divisible by 400 but is by 100, then the system outputs a no. If it's divisible by 4 but not by the other two numbers, it outputs that the year is a leap year. Finally, if it goes past those three if/else if statements, the default "else' will say the year is not a leap year.

For the random number generators, I used the built in "rand" function. To set the appropriate range, one must use the syntax "rand() % max + min". So, for the range of 1 to 6, I used "rand % 6 + 1". I used this same syntax to set the range of 1 to 20, and the range of 0 to 10,000.

Finally, for the "end program" option. This case statement will simply output that the program is ending, then return a Boolean value of true to a sentinel variable and cause the program loop to end.

# Testing plans

1. Test that the program can do factorial of
    a. 5
    b. 10
    c. 12
2. Test that the program can sum together
    a. 2+2
    b. 5+4+9
    c. 300+500
    d. 300+500+900+1000+580
3. Test that the program can multiply
    a. 2 * 2
    b. 4 * 8 * 9
    c. 4* 100,020
    d. 100 * 349 * 40 * 500
4. Test that the program can do
    a. $3^2$
    b. $9^5$
    c. $7^{10}$
5. Test the leap-year checker with known leap-years and known non-leap-years.
    a. 2020
    b. 2019
    c. 2016
    d. 2005
6. Test the random number generators
    a. 1-6 range, 5 times
    b. 1-20 range, 5 times
    c. 0-10,00 range, 5 times

# Test Results

| Testing Code | Expected Output | Program Output | Pass/Fail |
|---|---|---|---|
| 1A | 120 | 120 | P |
| 1B | 3,628,800 | 3,628,800 | P |
| 1C | 479,001,600 | 479,001,600 | P |
| 2A | 4 | 4 | P |
| 2B | 18 | 18 | P |
| 2C | 800 | 800 | P |
| 2D | 3,280 | 3,280 | P |
| 3A | 4 | 4 | P |
| 3B | 288 | 288 | P |
| 3C | 400,080 | 400,080 | P |
| 3D | 698,000,000 | 698,000,000 | P |
| 4A | 9 | 9 | P |
| 4B | 59,049 | 59,049 | P |
| 4C | 282,475,249 | 2.82475e+08. | P |
| 5A | Leap Year | Leap Year | P |
| 5B | Not Leap Year | Not Leap Year | P |
| 5C | Leap Year | Leap Year | P |
| 5D | Not Leap Year | Not Leap Year | P |
| 6A | (5 numbers 1 through 6) | 6, 6, 5, 5, 1, 2 | P |
| 6B | (5 numbers in range 1 through 20) | 5, 6, 6, 2, 12 | P |
| 6C | (5 numbers in range 0 through 10,000) | (2,995), (1,942), (4,827), (5,436), (2,391) | P |