Faculty of Engineering and Information Technology
School of Software

31927 - Applications Development with .NET
32998 - .NET Applications Development

SPRING 2018
ASSIGNMENT 1 SPECIFICATION

DUE DATE – Monday, 17th September 2018
This assignment is worth 25% of the total marks

Requirement Assignment Documentations

This assignment is split into two documents.

1. Assignment Specification. This document contains the specification for what you will be required to do.
2. Assignment Addendum. This document contains all other details regarding the assignment, including assignment submission and the assessment scheme.

Requirement

In this assignment you are to prepare a C# program that will act as a simple equation solver. The equation solver will be run from the command line and will only work with integer numbers, a single variable (X) and the following arithmetic operators: + - * / % ^2. The % operator is the modulus operator, not the percentage. ^2 is the square of a number/variable (e.g. X^2 = X^2)

For example, if the C# program is compiled to Equ.exe, the following demonstrates how it will appear in the command line:

calc 3X + 5 = 8
X = 1

In the command line, the argument is entered as the “calc” keyword followed by a space and followed by a sequence of numbers, variables, and operators, such as:
calc <Equation formula>
Hitting the enter key will cause the program to evaluate the arguments and print the result as one or more numbers separated by “,”:

X=<result>,<result>.

The console should remain open and allow the user to continue to enter equations to be solved.

The program must follow the math rules of solving equations with 1 variable type (i.e. linear and quadric style equations). For example:

2X – 1 = 0
3X + 8X = 33
4X - 7 = 8X - 5
X / (x+12) = 1 / 13
X^2 + X^2 + 4X + 9X + X = 10 + 1 * 2 % 4

The program must follow the usual laws of arithmetic which says:

1. The * / and % operators must all be evaluated before the + and – operators.
2. Operators must be evaluated from left to right.

For example, using Rule 1

X = 2 + 4 * 3 – 6
Becomes
X = 2 + 12 – 6
Which results in
X = 8

X^2 + 2X^2 = 27
Becomes
3X^2 = 27
Which results in X = 3

X^2 – 4 = 11 * 7
Becomes
X^2 – 4 = 77
Becomes
X^2 = 81
Which results in X = 9

If we did not use Rule 1 then 2+4*3–6 would become 6*3–6 and then 18–6 and finally 12. This is an incorrect result.
If we do not use Rule 2 then the following illustrates how it can go wrong

4*5%2

Going from left to right we evaluate the * first, which reduces the expression to 20%2 which becomes 0. If we evaluated the % first then the expression would reduce to 4*1 which becomes 1. This is an incorrect result. Remember, we are using integer mathematics when doing our calculations, so we get integer results when doing division. For example:

calc X = 20 / 4
X = 5

Also note that we can use the unary + and – operators. For example:

calc X = -5 / +2

X = -2

Your program must also check to make sure the command line argument is correct and handle the equation parts. If not your program must generate an appropriate error message and then terminate. For example:

calc X = - 3 + 4 / 6 (spaces represent splitters between numbers and operators)
Becomes X = -3 + 4 / 6

calc X = 3 + 4 / 6 + (space after+)
Becomes X = 3 + 4 / 6 + 0 (replace by 0)
Becomes X = 3 + 4 / 6

calc X = 3 + + + 4 - 6 (operation elimination)
Becomes X = 3 + 4 - 6

calc X = 3 - - - 8X - 4
Becomes X = 3 - 8X - 4

calc X + 3 + 6 + 3333333333333
Invalid Input (this is not equation and invalid as well)

calc X^2 = 2121212121 + 2020202020
Out of Integer Range

calc X = 4 % 0
calc X = 4 / 0
Division by zero

Program Hints

1. Getting your program to solve expressions that only use the + and – operators is fairly easy. I would suggest you get your program working at this level before attempting to get it to work with the other operators.

2. You will find this problem much easier to solve if you application has a good design concept; create a generic input that takes all parameters and operations and break it down into smaller methods responsible for validations, aggregations and calculations. For example, enumerate the possible operations and then have a method that solves (number operator number). Also have a method that gets the number from a string and another that gets the operator from a string. Also note that splitting your code up will give you a better design mark.

Assignment Objectives

The purpose of this assignment is to demonstrate competence in the following skills.

- Problem solution
- Employ basic mathematic skills to develop C# program
- Program design
- Array and string manipulation
- Command line arguments
- Creating Object Oriented methods in C#

These tasks reflect all the subject objectives.

The solution of this assignment will take between 200 and 300 lines of code, including white space (blank lines) comments, etc. The exact number will depend on how the students solve the task. As part of your subject workload assessment, it is estimated this assignment will take more than 30 hours to complete.