**Design Document Draft**

**Requirements:**

Requirement 1 will be met by creating a front panel Boolean button for each operation of the calculator.

Requirement 2 will be met by having a 10-character string indicator output box. Having a larger display than 6 will allow the user to input more operators and operands, which will make the calculator easier to use for more complex calculations.

Requirement 3 will be met by using double precision floating point data storage for all floating-point numbers, 16 bit signed integer for any integer numbers, and unsigned 8 bit integers for operators.

Requirement 4 will also be met by using double precision floating point data types for floating point number storage.

Requirement 5 will be met by using a single string output indicator display box (showing 10 characters from req 2)

Requirement 6 will be met by adding the next input as a character to the end of the current displayed output string. Any new button press will activate a case statement. If the button represents an operand, the number will be stored in a shift register so it can be shown on the display later. If it’s an operator, the operation will be performed and stored, and the operator will be stored for output on the display.

Requirement 7 will be met by including each of those 10 boolean control buttons on the front panel.

Requirements 8-11 will be met by adding their boolean control buttons labelled +, -, x, ÷, respectively. These will store the operator in a shift register so it can be shown on the display and switch the appropriate case statements to perform the right operation on the current inputs. This (nonfinal) floating point answer is stored in a shift register (called “result”, for example) for the next calculation.

Requirement 12 will be met by adding a boolean button labelled . on the front panel. This will increment a shift register by 1, showing that the next entered value is meant to be divided by 10. If the shift register is 2, then divide by 100 etc. This will reset to 0 when the CLEAR, =, or another operator is pressed so the next value is not divided.

Requirement 13 will be met by adding a boolean button labelled = on the front panel. This will switch a case statement so the string output display only takes 1 input, from the value in the shift register “result” and convert it from a floating point to a string.

Requirements 14 and 15 will be met by adding a boolean button labelled CLEAR on the front panel. This will set the “result” shift register to 0.00 and switch case statement so the display reads only from it (just like the = button). This will also reset every shift register to their default value when the program is first run. This would include for example the shift register used for a decimal point.

Requirement 16 will be met by adding a boolean button labelled DELETE on the front panel. This will delete the last entered (rightmost) character on the string display box by clearing the shift register storing the last saved operator/operand. This means the display won’t see the last thing entered.

Requirement 17 will be met by adding 6 or more boolean buttons labelled appropriately on the front panel. When pressed, these will switch a case statement to input the floating-point constant (which are constants defined in the block diagram).

Requirement 18 will be met by converting the shift register “result” literally, to a string when = is pressed.

Requirement 19 will be met by detecting two non – sign operators and the first has no second operator and the second has no first operator, the first operator will pass the value through and perform no actions. This allows the second operator to use the nearest left-hand number as an input.

Requirement 20 will be met by detecting 1 operand and no other operators or operands (end of inputs) and copying the operand into the other input wire of the operator function. We can check this by using a shift register and checking if the operand is equal to one and if so to duplicate the operand or multiply by the same number.

Requirements 21, 22 are automatically done by LabView.

Requirement 23 the calculator will display NaN when dividing 0 by 0 which is automatically done by LabVIEW

**Overall Approach**

The calculator will include a set of labelled Boolean buttons on the front panel. When pressed, these will store the button pressed for use when the next button is pressed. It will use shift registers to store operators and floating point result values. These will be used in the next “loop” of operations. The front panel string indicator will continuously update to display all operands and operators in the correct order for the user to see their inputs. After the = button on the front panel is pressed, the calculator will clear the display, perform all operations, and then output the result into the same string display box.

**Materials Needed:**

1 windows computer with LabView installed and an internet connection

2 students with some experience in LabView programming

At least 1 professor or TA who is capable at programming in LabView

**Schedule:**

Concept Doc 9/7/18

Requirements Doc 9/21/18

Design Doc 10/26/18

GUI Design 10/26/18

Functional Blk Diagram 11/9/18

Module Design 11/9/18

GUI Operational 11/16/18

Functional Prototype 11/16/18

Prototype Review 11/20/18

Test Plan 11/23/18

Test Results 11/30/18

Implementation Log 12/4/18

Final Project Due 12/7/18

**Areas of Risk and "Plan-B" if needed:**

I see no obvious health and safety concerns for an entirely software-based project. As a result, no plan-Bs have been decided.

**Implementation Log:**