

Lab 7

ECE 1574

Spring 2006

Background: When you are at school you learn that different arithmetic operators have different precedence, although that may not be the exact phrase the teacher uses. You are probably given a mnemonic such as "Please Excuse My Dear Aunt Sally" (meaning Parentheses, Exponentiation (roots and powers), Multiplication, Division, Addition, Subtraction) to tell you the order in which to do sums. For example, if you are given:

$$1 + 2 \times 3 = ?$$

you know to multiply 2 by 3 before adding the 1. Early pocket calculators were not acquainted with Aunt Sally, and would evaluate this expression as 9. This type of notation is called 'infix' notation because the operators lie in between the operands. Another type of notation, called Reverse Polish Notation (RPN) can be used to alleviate the need for an order of operations and parentheses. RPN is a scheme where the operators follow the operands and thus utilize a 'postfix' notation. This is an advantage because the operators appear in the order required for computation. For example, in order to ensure the above expression is evaluated as 9, we might write it as:

$$(1 + 2) \times 3 = ?$$

But, in RPN the same expression can be written as:

$$3 \ 2 \ 1 + \times$$

Both expressions are evaluated in the same way, but the RPN notation removes the need for the parentheses. Another example is:

$$((1 + 2) \times 4) + 3 = ? \quad [\text{infix notation}] \quad \text{Result} = 15$$

$$1 \ 2 + 4 \times 3 + \quad [\text{RPN or postfix notation}] \quad \text{Result} = 15$$

The best way to visualize RPN expressions is to imagine a stack that stores each of the operands. As each operand is entered, it is pushed onto the stack. When an operator is entered, the two values at the top of the stack are popped off of the stack, evaluated using the operator, and the result is pushed back onto the stack. At the end of the computation, the result is the only operand on the stack.

While a stack is the best way to implement this design, a simple array can be used as well as long as we keep track of the amount of data stored in the array. This is how you will implement your RPN calculator. You will have an array called "stack_array" that will hold double values (they have to be doubles since a division might result in a non-

integer answer). Assume that the `stack_array` can hold a maximum of 8 double values. In addition you will use an integer variable “`top_of_stack`” to keep track of the number of values that are stored in the array. Initially `top_of_stack` will be set to 0 since there will be no values stored in the array. When you *push* and *pop* values onto the array, you will increment or decrement the value of `top_of_stack`, respectively. For example, if we wanted to do the expression $3\ 2\ 1 + x$, the array would look like:

Initial Condition:

--	--	--	--	--	--	--	--

top_of_stack = 0 (no elements in the array)

LCD[line 0] = “RPN Calculator!”

Enter 3:

3							
---	--	--	--	--	--	--	--

top_of_stack = 1 (3 is pushed on array)

LCD[line 0] = “3”

Enter 2:

3	2						
---	---	--	--	--	--	--	--

top_of_stack = 2 (2 is pushed on array)

LCD[line 0] = “3 2”

Enter 1:

3	2	1					
---	---	---	--	--	--	--	--

top_of_stack = 3 (1 is pushed on array)

LCD[line 0] = “3 2 1”

Enter +:

3	3						
---	---	--	--	--	--	--	--

top_of_stack = 2 (2 & 1 popped off array and 3 (2 + 1) is pushed back on)

LCD[line 0] = “3 2 1 +”

Enter x:

9							
---	--	--	--	--	--	--	--

top_of_stack = 1 (3 & 3 popped off array and 9 (3 x 3) is pushed back on)

LCD[line 0] = “3 2 1 + x”

LCD[line 1] = “Result = 9”

Your job in this project is to create an RPN calculator using your Fox11 board. The input will be the keypad and the results will be displayed on the LCD screen. The LED’s will show the state of the array. You will implement addition, subtraction, multiplication, and division.

Specifications:

- (1) At the beginning of the program line 0 of the LCD should read "RPN Calculator!"
- (2) 0-9 on the keypad represent the operand inputs and A-D represent the operators. A = addition, B = subtraction, C = multiplication, and D = division. E is the enter key. The user will input an expression using the keypad and press enter (E) after each operand or operator is entered. Once the entire expression has been entered, F is pressed to signal that the user is done entering values.
- (3) Line 0 of the LCD will contain the string of operators and operands in the order they are entered (postfix) and will update after each value is entered. So, using the above example, the LCD should display 3 by itself after 3 is entered, then 3 2 after the two is entered, then 3 2 1 after the 1 is entered, and so on.
- (4) The LED's will represent the status of `top_of_stack`. If `top_of_stack` is 1, LED 0 should be lit. If there are 2 elements in the array, LED's 0 and 1 should be lit. This assumes that there will be no more than 8 operands in the array at one time.
- (5) When the user presses F the final time to signify that they are done entering values, line 1 of the LCD should read "Result = " followed by the result. LED 0 should be lit, signifying that the one value left on the array is the result.
- (6) The program should then loop and allow the user to enter a new expression.

Note:

When reading from the keypad, you must poll for the user to input a number. You cannot use the `serial_getline` function and press enter on the computer keyboard to get the pressed keypad value as in previous projects.

Program Validation

TA initials:

- (1) The program initially displays “RPN Calculator” on line 0 of the LCD.
- (2a) Press ‘1’ and then ‘E’ on the keypad. Line 0 of the LCD should display a ‘1’. LED 0 should light up.
- (2b) Press ‘2’ and then ‘E’ on the keypad. Line 0 of the LCD should display ‘1 2’. LED’s 0 and 1 should light up.
- (2c) Press ‘A’ and then ‘E’ on they keypad. Line 0 of the LCD should display ‘1 2 +’. LED 0 should light up.
- (2d) Press ‘4’ and then ‘E’ on they keypad. Line 0 of the LCD should display ‘1 2 + 4’. LED’s 0 and 1 should light up.
- (2e) Press ‘C’ and then ‘E’ on they keypad. Line 0 of the LCD should display ‘1 2 + 4 x’. LED 0 should light up.
- (2f) Press ‘3’ and then ‘E’ on the keypad. Line 0 of the LCD should display ‘1 2 + 4 x 3’. LED’s 0 and 1 should light up.
- (2g) Press ‘B’ and then ‘E’ on the keypad. Line 0 of the LCD should display ‘1 2 + 4 x 3 –’. LED 0 should light up.
- (2h) Press ‘3’ and then ‘E’ on the keypad. Line 0 of the LCD should display ‘1 2 + 4 x 3 – 3’. LED’s 0 and 1 should light up.
- (2i) Press ‘D’ and then ‘E’ on the keypad. Line 0 of the LCD should display ‘1 2 + 4 x 3 – 3 /’. LED 0 should light up.
- (3) Press ‘F’ on they keypad. Line 0 of the LCD should display ‘1 2 + 4 x 3 – 3 /’. LED 0 should light up. Line 1 of the LCD should display ‘3’.
- (4) The program should loop so that the user can enter a new sequence of inputs without the program needing to be restarted.

1)_____

2)_____

3)_____

4)_____

Once you have completed each of the above steps, please validate your lab by showing the TA that all five functions work properly. The TA should initial each step above and then sign below to show that the lab has been validated. This lab (and all further labs) must be validated during your lab section.

Your Name

Date

GTA name / Signature