Ethan Axl S. Burayag Group 12
Ezra Jeonadab G. Del Rosario
S14 - CCDSALG

Project 1 - Expression Evaluation (Stack and Queue Application)

Documentation

Brief description of the implementation of stacks and queues

We used structures to model stacks and queues. The reasoning is that structs allow for attributes and information to be grouped and easily accessed. It is also the closest thing C has for modeling reference types. We used one-dimensional arrays to hold the collection of elements inside the stack/queue. To adhere to the specification, we made two separate implementations for integers and string elements. The stacks were used to hold the operands and operators. For queues, we made a circular implementation with character elements. The queue contained the equivalent postfix expression of the infix input and will later be used for evaluation. For the different operations, we defined specific functions for stacks and queues, accommodating the data type they take in. We also restricted access to add and remove functions (dependent on the type).

Brief description of the algorithms (infix to postfix, postfix evaluation)

We used the string stack and the character queue as the main data structures to convert from infix to postfix algorithm. We followed the sequence of the algorithm taught, wherein the operators are stored in the string stack and popped into the character queue, considering their precedence and associativity; the operands, however, are directly stored in the character queue. Moreover, we employed loops to catch more than one digit or more than one character operands and operators, such as relational and logical operators. Also, separate functions were defined to determine operator precedence and associativity, returning an integer based on priority.

We used the integer stack and the resulting character queue to evaluate the postfix expressions. We also adhered to the general sequence of the algorithm, which involves the continuous pushing of operands to the stack whenever new operations are conducted until the result is obtained. We used loops to create substrings for operands, to be converted to integers, and substrings for the operator. In addition, we utilized a function to check if an element is an integer, as well as a function that returns the integer equivalent of the string operand. This is so that integers are properly identified and converted for evaluation. With that said, we defined functions that evaluate unary and non-unary expressions.

Limitations

Aside from the limitations in the specification, our program is also limited in data range. The maximum nonnegative integer it could hold is the upper limit of the integer data type. Anything exceeding that will cause an overflow.

Distribution of tasks

- Ethan Axl Burayag - Infix to Postfix Algorithm
- Ezra Jeonadab Del Rosario - Postfix Evaluation

Precedence Table

Operator	Type	Associativity
()	Grouping	left-to-right
!	Unary Operator	right-to-left
^	Arithmetic Operator	right-to-left
*, /, %	Arithmetic Operator	left-to-right
+, -	Arithmetic Operator	left-to-right
>, <, >=, <=	Relational Operator	left-to-right
==, !=	Relational Operator	left-to-right
&&	Logical Operator	left-to-right
||	Logical Operator	left-to-right

Test Cases

- Test cases are grouped into increasing difficulty.

Legend:

- P: Postfix expression
- E: Evaluated result

Easy

Test Description	Sample Input	Expected Output	Actual Output	P/F
Expressions are made up of simple arithmetics (+, -, *, /, %) with groupings to show explicit precedence	3+4*2	P: 3 4 2 * + E: 11	P: 3 4 2 * + E: 11	P
	3+5/2-1	P: 3 5 2 / + 1 - E: 4	P: 3 5 2 / + 1 - E: 4	
	1+2*3*4	P: 1 2 3 * 4 * + E: 25	P: 1 2 3 * 4 * + E: 25	
	5%2*8-6	P: 5 2 % 8 * 6 - E: 2	P: 5 2 % 8 * 6 - E: 2	
	8%2-10	P: 8 2 % 10 - E: -10	P: 8 2 % 10 - E: -10	
	9/(3+1)	P: 9 3 1 + / E: 2	P: 9 3 1 + / E: 2	
	3+4*2/(1-5)	P: 3 4 2 * 1 5 - / + E: 1	P: 3 4 2 * 1 5 - / + E: 1	
	2+3*(4-1)	P: 2 3 4 1 - * + E: 11	P: 2 3 4 1 - * + E: 11	
	7+(5-2)*3	P: 7 5 2 - 3 * + E: 16	P: 7 5 2 - 3 * + E: 16	
	(6/3)+5*2	P: 6 3 / 5 2 * + E: 12	P: 6 3 / 5 2 * + E: 12	

Medium

Test Description	Sample Input	Expected Output
Expressions are made up of simple arithmetics (+, -, *, /, %) with groupings, relational operators (>,<,>=,<=,==,!=), and logical operators (!, &&, ||)	7*(5-3)!=4||2>1	P: 7 5 3 - * 4 != 2 1 > || E: 1
	(8/2)==4&&6<7	P: 8 2 / 4 == 6 7 < && E: 1
	5*2+3>=13||!(4-2>1)	P: 5 2 * 3 + 13 >= 4 2 - 1 > ! || E: 1
	3*(2+4)/2>7&&9!=10	P: 3 2 4 + * 2 / 7 > 9 10 != && E: 1
	(5+3)==8||10%3==1	P: 5 3 + 8 == 10 3 % 1 == || E: 1
	4*2/1-3<=5&&8>=8	P: 4 2 * 1 / 3 - 5 <= 8 8 >= && E: 1
	7%2+3>4||6/3==2	P: 7 2 % 3 + 4 > 6 3 / 2 == || E: 1
	2*(3+4)==14&&5>1	P: 2 3 4 + * 14 == 5 1 > && E: 1
	(8/4)==2||5-3!=2	P: 8 4 / 2 == 5 3 - 2 != || E: 1
	6*2+1==13&&9>8	P: 6 2 * 1 + 13 == 9 8 > && E: 1
	10-(3*2)!=4&&7+2<=10	P: 10 3 2 * - 4 != 7 2 + 10 <= && E: 0
	(4+5)/3==3||8%2==0	P: 4 5 + 3 / 3 == 8 2 % 0 == || E: 1
	5*(6/2)<20&&7-2>4	P: 5 6 2 / * 20 < 7 2 - 4 > && E: 1
	3+4*2<=11&&6%3==0	P: 3 4 2 * + 11 <= 6 3 % 0 == && E: 1
	2*(3+5)==16||7<2	P: 2 3 5 + * 16 == 7 2 < || E: 1
	(4/2)+7!=9&&8>=8	P: 4 2 / 7 + 9 != 8 8 >= &&

	9-3*2==3||5<=5 8/4+3!=5&&2*3==6 5+(6*2)>15||4==4 10+(3*2)<=15&&7%2==1	E: 0 P: 9 3 2 * - 3 == 5 5 <= || E: 1 P: 8 4 / 3 + 5 != 2 3 * 6 == && E: 0 P: 5 6 2 * + 15 > 4 4 == || E: 1 P: 10 3 2 * + 15 <= 7 2 % 1 == && E: 0

Continuation

Actual Output	P/F
P: 7 5 3 - * 4 != 2 1 > || E: 1 P: 8 2 / 4 == 6 7 < && E: 1 P: 5 2 * 3 + 13 >= 4 2 - 1 > ! || E: 1 P: 3 2 4 + * 2 / 7 > 9 10 != && E: 1 P: 5 3 + 8 == 10 3 % 1 == || E: 1 P: 4 2 * 1 / 3 - 5 <= 8 8 >= && E: 1 P: 7 2 % 3 + 4 > 6 3 / 2 == || E: 1 P: 2 3 4 + * 14 == 5 1 > && E: 1 P: 8 4 / 2 == 5 3 - 2 != || E: 1	P

P: 6 2 * 1 + 13 == 9 8 > &&
E: 1

P: 10 3 2 * - 4 != 7 2 + 10 <= &&
E: 0

P: 4 5 + 3 / 3 == 8 2 % 0 == ||
E: 1

P: 5 6 2 / * 20 < 7 2 - 4 > &&
E: 1

P: 3 4 2 * + 11 <= 6 3 % 0 == &&
E: 1

P: 2 3 5 + * 16 == 7 2 < ||
E: 1

P: 4 2 / 7 + 9 != 8 8 >= &&
E: 0

P: 9 3 2 * - 3 == 5 5 <= ||
E: 1

P: 8 4 / 3 + 5 != 2 3 * 6 == &&
E: 0

P: 5 6 2 * + 15 > 4 4 == ||
E: 1

P: 10 3 2 * + 15 <= 7 2 % 1 == &&
E: 0

Difficult

Test Description	Sample Input	Expected Output
Expressions are made up of arithmetics (+, -, *, /, %, ^) with groupings, relational operators (>,<,>=,<=,==,!=) and logical operators (!, &&, ||). Cases with division by zero are now also included.	9^(3+1)	P: 9 3 1 + ^ E: 6561
	3+!(4*2/(1-5))	P: 3 4 2 * 1 5 - / ! + E: 3
	!(2*3^(4-1))>10&&5<=6	P: 2 3 4 1 - ^ * ! 10 > 5 6 <= && E: 0
	7*!(5-3)!=4||2>1	P: 7 5 3 - ! * 4 != 2 1 > || E: 1
	!(8/2)==4&&6<7	P: 8 2 / ! 4 == 6 7 < && E: 0
	5*!(2+3)>=13||!(4-2>1)	P: 5 2 3 + ! * 13 >= 4 2 - 1 > ! || E: 0
	!(3*(2+4)/2>7)&&9!=10	P: 3 2 4 + * 2 / 7 > ! 9 10 != && E: 0
	!(5+3)==8||10%3==1	P: 5 3 + ! 8 == 10 3 % 1 == || E: 1
	4*2/!(1-3)<=5&&8>=8	P: 4 2 * 1 3 - ! / 5 <= 8 8 >= && E: Division by zero error!
	7%!(2+3)>4||6/3==2	P: 7 2 3 + ! % 4 > 6 3 / 2 == || E: Division by zero error!
	2*!(3+4)==14&&5>1	P: 2 3 4 + ! * 14 == 5 1 > && E: 0
	!(8/4)==2||5-3!=2	P: 8 4 / ! 2 == 5 3 - 2 != || E: 0
	6*!(2+1)==13&&9>8	P: 6 2 1 + ! * 13 == 9 8 > && E: 0
	10-!(3*2)!=4&&7+2<=10	P: 10 3 2 * ! - 4 != 7 2 + 10 <= && E: 1
	!(4+5)/3==3||8%2==0	P: 4 5 + ! 3 / 3 == 8 2 % 0 == || E: 1
	5*!(6/2)<20&&7-2>4	P: 5 6 2 / ! * 20 < 7 2 - 4 > &&

		E: 1
	3+!(4*2)<=11&&6%3==0	P: 3 4 2 * ! + 11 <= 6 3 % 0 == && E: 1
	2*!(3+5^2)==16||7<2	P: 2 3 5 2 ^ + ! * 16 == 7 2 < || E: 0
	(4/2)+7!=!(9)&&8>=8	P: 4 2 / 7 + 9 ! != 8 8 >= && E: 1
	9-!(3*2)==3||5<=5	P: 9 3 2 * ! - 3 == 5 5 <= || E: 1
	8/!(4+3)+3!=5&&2*3==6	P: 8 4 3 + ! / 3 + 5 != 2 3 * 6 == && E: Division by zero error!
	1+(4^2^2)/8	P: 1 4 2 2 ^ ^ 8 / + E: 33

Continuation

Actual Output	P/F
P: 9 3 1 + ^ E: 6561 P: 3 4 2 * 1 5 - / ! + E: 3 P: 2 3 4 1 - ^ * ! 10 > 5 6 <= && E: 0 P: 7 5 3 - ! * 4 != 2 1 > || E: 1 P: 8 2 / ! 4 == 6 7 < && E: 0 P: 5 2 3 + ! * 13 >= 4 2 - 1 > ! || E: 0 P: 3 2 4 + * 2 / 7 > ! 9 10 != && E: 0	P

P: 5 3 + ! 8 == 10 3 % 1 == || E: 1 P: 4 2 * 1 3 - ! / 5 <= 8 8 >= && E: Division by zero error! P: 7 2 3 + ! % 4 > 6 3 / 2 == || E: Division by zero error! P: 2 3 4 + ! * 14 == 5 1 > && E: 0 P: 8 4 / ! 2 == 5 3 - 2 != || E: 0 P: 6 2 1 + ! * 13 == 9 8 > && E: 0 P: 10 3 2 * ! - 4 != 7 2 + 10 <= && E: 1 P: 4 5 + ! 3 / 3 == 8 2 % 0 == || E: 1 P: 5 6 2 / ! * 20 < 7 2 - 4 > && E: 1 P: 3 4 2 * ! + 11 <= 6 3 % 0 == && E: 1 P: 2 3 5 2 ^ + ! * 16 == 7 2 < || E: 0 P: 4 2 / 7 + 9 ! != 8 8 >= && E: 1 P: 9 3 2 * ! - 3 == 5 5 <= || E: 1 P: 8 4 3 + ! / 3 + 5 != 2 3 * 6 == && E: Division by zero error! P: 1 4 2 2 ^ ^ 8 / + E: 33	

Very Difficult

Test Description	Sample Input	Expected Output
Expressions are made up of arithmetics (+, -, *, /, %, ^) with groupings, relational operators (>,<,>=,<=,==,!=) and logical operators (!, &&, ||). The expression provided consists of 255 characters.	((5>3)*(12+8||4-4))+!(16+4+2)||(9!=8)+(2<3*(8/4))+((10!=5+3*2)&&((3-4)||!(6>=7)&&(8+8*2))||(2==2)&&(9>1))==((4+5+5-9)&&!(6*2<15-4||7!=7)+(8-8&&10>9))+(((5<=10)&&(3>1||4+4))&&!(6<2)||(7*7)&&(9+9||1+3>0))==((5>3)&&(12<=8||4==4))&&!(16==4)||(9<8)+(7+3*3)+5-2	P: 5 3 > 12 8 + 4 4 - || * 16 4 + 2 + ! + 9 8 != 2 3 8 4 / * < + 10 5 3 2 * + != 3 4 - 6 7 >= ! 8 8 2 * + && || && 2 2 == 9 1 > && || + 4 5 + 5 + 9 - 6 2 * 15 4 - < 7 7 != || ! 8 8 - 10 9 > && + && 5 10 <= 3 1 > 4 4 + || && 6 2 < ! && 7 7 * 9 9 + 1 3 + 0 > || && || + == 5 3 > 12 8 <= 4 4 == || && == 16 4 == ! && || 9 8 < 7 3 3 * + + 5 + 2 - || E: 1

Continuation

Actual Output	P/F
P: 5 3 > 12 8 + 4 4 - || * 16 4 + 2 + ! + 9 8 != 2 3 8 4 / * < + 10 5 3 2 * + != 3 4 - 6 7 >= ! 8 8 2 * + && || && 2 2 == 9 1 > && || + 4 5 + 5 + 9 - 6 2 * 15 4 - < 7 7 != || ! 8 8 - 10 9 > && + && 5 10 <= 3 1 > 4 4 + || && 6 2 < ! && 7 7 * 9 9 + 1 3 + 0 > || && || + == 5 3 > 12 8 <= 4 4 == || && == 16 4 == ! && || 9 8 < 7 3 3 * + + 5 + 2 - || E: 1	P