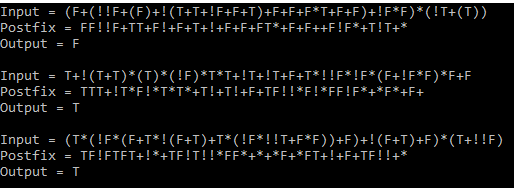
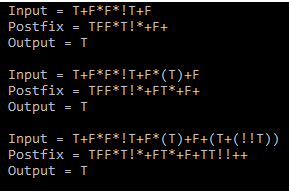
Juan Bermudez  
11/05/2018   
CMPE 3333

Read Me:   
The program has the functions and their descriptions in the header file EvalBool.h. The file project4.cpp contains the main function where the program is executed. The program was executed using visual studios, and using the C++ STL template for stacks.

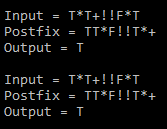
Testing Cases for proof that the postfix evaluation of Boolean characters is correct:

Original test runs on Project 4 worksheet evaluated  


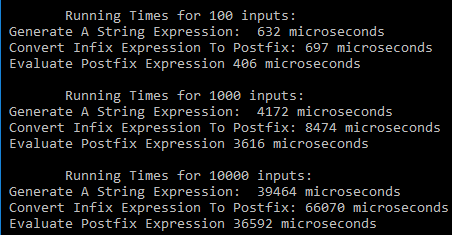


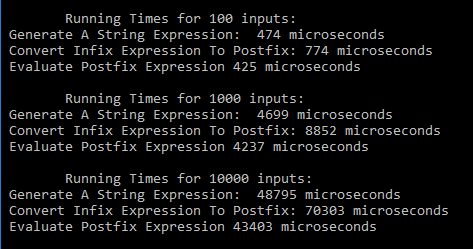


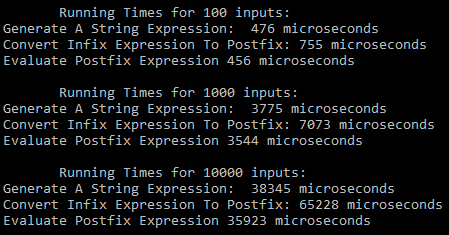
Testing for smaller cases shows that the EvaluateBooleanPostfix function correctly evaluates the operands based on precedence.

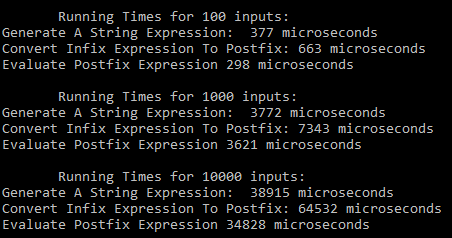


Now that the evaluations have shown to be correct, the running times for larger input sizes can be evaluated. Below are the running times for 3 different sets of trials of input sizes 100, 1000, and 10000:







  
  
Data:

Running Times in microseconds for

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Input Size (N)** | **Trial 1** | | **Trial 2** | | **Trial 3** | | **Trial 4** | |
|  | **Infix To Postfix** | **Postfix  Evaluation** | **Infix To Postfix** | **Postfix  Evaluation** | **Infix To Postfix** | **Postfix  Evaluation** | **Infix To Postfix** | **Postfix  Evaluation** |
| **100** | 697 µs | 406 µs | 774 µs | 425 µs | 755 µs | 456 µs | 663 µs | 298 µs |
| **1000** | 8474 µs | 3616 µs | 8852 µs | 4237 µs | 7073 µs | 35923µs | 7343 µs | 3621µs |
| **10000** | 66070 µs | 36592 µs | 70303 µs | 43403 µs | 65228 µs | 35923µs | 64532 µs | 34828 µs |

Report:   
  
 An infix notation reads an expression from left to right and performs an evaluation on the operands based on the precedence of the operators. The operators are usually in between the operands. For this reason, infix expressions are usually harder to evaluate for computers because they must decide on the precedence of the operators before they perform a calculation, and so it takes a longer time.  
 This same way of calculating an infix expression is represented by the InfixToPostfix function, where a Boolean expression is converted to its equivalent postfix. Thus, the function will evaluate the expression based on the order of operations, and then output the solution, where the precedence of the operators are placed in the right hand side of the expression to make a postfix notation. For this reason, it may be said that the InfixToPostfix function simulates an infix calculation. The running time of this function is of O(N).  
  
 Total: N\*O(N)(for loop) + O(8) (if else statements) = O(N)   
  
 Postfix notation is a lot easier for computers to process because the order of operations is not needed to evaluate the expression. In postfix notation, the operands are pushed to a stack, and when there is an operator, then the process for that operator evaluates the two elements from the stack, and the result is pushed back into the stack. Hence, postfix notation does not deal with precedence as an infix notation does. The process is much more straightforward, simpler, and faster.  
 The EvaluateBooleanPostfix function will take the postfix string, evaluate the boolean expression and return a char for the result. The running time for this function is O(N).   
  
 Total: N\*O(N) (for loop) + O(2)(if statements) = O(N)

Results:   
Even if both expressions are of O(N) running time, the differences in efficiency between infix and postfix notations are very clear by the results. In every trial, the time to evaluate a postfix expression was quicker by more than half the time of an infix evaluation. For example, an infix calculation took 6948 microseconds to process, and the postfix calculation took 3592 microseconds. Thus, the actual results are corresponded with the theoretical analysis.