Name : Stack, Algebraic Expressions, CSE 274

* Use a stack to check whether the string/expression “{ [ ( ) ] }” is balanced or not:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Rule: Opening ( or { or [  push to stack  Closing ( or { or [  pop from stack |  |  |  |  |  |  |
| { | [ | ( | ) | ] | } |

* Use a stack to check whether the string/expression “[ ( ) ] }” is balanced or not:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule: Opening ( or { or [  push to stack  Closing ( or { or [  pop from stack |  |  |  |  |  |
| [ | ( | ) | ] | } |

* Use a stack to check whether the string/expression “{ [ ( ) ] ” is balanced or not:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule: Opening ( or { or [  push to stack  Closing ( or { or [  pop from stack |  |  |  |  |  |
| { | [ | ( | ) | ] |

* Convert the infix expression “a + b \* c” to postfix form:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule: 1. Operand: Append to output  2. Operator ^: Push 3.Operator: +-\*/: Pop operators until the stack is empty or top has a lower precedence then the new operator. Push new operator 4.Open parenthesis: Push 5.Close parenthesis: Pop operators and append to output until an open parenthesis popped. Discard both parentheses. |  |  |  |  |  |
| a | + | b | \* | c |
| **Output:** |  | | | | |

* Convert the infix expression “a - b + c” to postfix form:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule: 1. Operand: Append to output  2. Operator ^: Push 3.Operator: +-\*/: Pop operators until the stack is empty or top has a lower precedence then the new operator. Push new operator 4.Open parenthesis: Push 5.Close parenthesis: Pop operators and append to output until an open parenthesis popped. Discard both parentheses. |  |  |  |  |  |
| a | - | b | + | c |
| **Output:** |  | | | | |

* Convert the infix expression “a ^ b ^ c” to postfix form:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rule: 1. Operand: Append to output  2. Operator ^: Push 3.Operator: +-\*/: Pop operators until the stack is empty or top has a lower precedence then the new operator. Push new operator 4.Open parenthesis: Push 5.Close parenthesis: Pop operators and append to output until an open parenthesis popped. Discard both parentheses. |  |  |  |  |  |
| a | ^ | b | ^ | c |
| **Output:** |  | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| a | / | b | \* | ( | c | + | ( | d | - | e | ) | ) |
| **Output:** | | | | | | | | | | | | |

* Convert the infix expression “a / b \* (c + (d - e) )” to postfix form:
* Evaluate postfix expression, a b + c /. Here a = 2. b = 4, and c = 3. [Using a single stack]

The equivalent infix expression is ( a + b ) / c

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Rule: 1. Operand: Push  2. Operator: Pop twice, first one right side of the operator and second one left side of the operator. Perform the operation and push back the result to stack. |  |  |  |  |  |  |  |
| a | b | + |  | c | / |  |
| **Output:** |  | | | | | | |

* Evaluate infix expression, a + b \* c or 2 + 3 \* 4. [Using two stacks]

Scan the expression and push to stack:  
(maintaining precedence of the operators)

Use two stacks, one for operators and one for operands

Operators Operands

Pop one operator and two operands. Evaluate and then push back

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |