##### **Transform each of the following expressions to prefix and postfix**

1. A+B-C

Prefix A+B-C

=+AB-C

=-+ABC

Postfix A+B-C

=AB+-C

=AB+C-

2. (A+B)\*(C-D)$E\*F

Postfix (A+B)\*(C-D)$E\*F

=(AB+)\*(CD-)$E\*F

=(AB+)\*(CD-)E\*F$

=AB+CD-\*EF\*$

Prefix (A+B)\*(C-D)$E\*F

=(+AB)\*(-CD)$E\*F

=$(+AB)\*(-CD)E\*F

=$\*+AB-CD\*EF

3. (A+B)\*(C$(D-E)+F)-G

Postfix (A+B)\*(C$(D-E)+F)-G

=(AB+)\*(C$(DE-)+F)-G

=(AB+)\*(CDE-$+F)-G

=(AB+)\*(CDE-$F+)-G

=AB+CDE-$F+\*-G

=AB+CDE-$F+\*G-

Prefix (A+B)\*(C$(D-E)+F)-G

=(+AB)\*(C$(-DE)+F)-G

=(+AB)\*(+$C-DEF)-G

=\*+AB+$C-DEF-G

=-\*+AB+$C-DEFG

4. A+(((B-C)\*(D-E)+F)/G)$(H-J)

Postfix A+(((B-C)\*(D-E)+F)/G)$(H-J)

=A+((BC-)\*(DE-)+F)/G)$(H-J)

=A+(BC-DE-\*+F)/G)$(H-J)

=A+(BC-DE-\*F+)/G)$(H-J)

=A+(BC-DE-\*F+G/)$(HJ-)

=A+BC-DE-\*F+G/HJ-$

=ABC-DE-\*F+G/HJ-$+

Prefix A+(((B-C)\*(D-E)+F)/G)$(H-J)

=A+((-BC)\*(-DE)+F)/G)$(H-J)

=A+(\*-BC-DE+F)/G)$(H-J)

=A+(+\*-BC-DEF)/G)$(H-J)

=A+(/+\*-BC-DEFG)$(-HJ)

=A+$/+\*-BC-DEFG-HJ

=+A$/+\*-BC-DEFG-HJ

##### **Transform each of the following prefix expressions to infix**

1. +-ABC

Prefix

=+(A-B)C

=(A-B)+C

2. +A-BC

=+A(B-C)

=A+(B-C)

3. ++A-\*$BCDI +EF\*GHI

= ++A-\*$BCDI +EF(G\*H)I

=++A-\*$BCDI (E+F)(G\*H)I

=++A-\*(B$C)DI (E+F)(G\*H)I

=++A-((B$C)\*D)I (E+F)(G\*H)I

=++A(((B$C)\*D)-I) (E+F)(G\*H)I

=+(A+(((B$C)\*D)-I) (E+F))(G\*H)I

=((A+(((B$C)\*D)-I)+ (E+F)))(G\*H)I

4. +-$ABC\*D\*\*EFG

=+-$ABC\*D\*(E\*F)G

=+-$ABC\*D((E\*F)\*G)

=+-$ABC(D\*((E\*F)\*G))

=+-(A$B)C(D\*((E\*F)\*G))

=+((A$B)-C)(D\*((E\*F)\*G))

=((A$B)-C)+(D\*((E\*F)\*G))

##### **Transform each of the following postfix expressions to infix**

1. AB+C-

=A+BC-

=A+B-C

2. ABC+-

=A(B+C)-

=A-(B+C)

=A-(B+C)

3. AB-C+DEF-+$

=(A-B)C+DEF-+$

=((A-B)+C)DEF-+$

=((A-B)+C)D(E-F)+$

=((A-B)+C)(D+(E-F))$

=((A-B)+C)$(D+(E-F))

4. ABCDE-+$\*EF\*-

=ABC(D-E)+$\*EF\*-

=AB(C+(D-E))$\*EF\*-

=A(B$(C+(D-E)))\*EF\*-

=(A\*(B$(C+(D-E))))EF\*-

=(A\*(B$(C+(D-E))))(E\*F)-

=(A\*(B$(C+(D-E))))-(E\*F)

##### **Apply the evaluation algorithm in the text to evaluate the following postfix expressions Assume A=1,B=2,C=3**

1. AB+C-BA+C$-

Here the given postfix expression is

AB+C-BA+C$-

And the value is **A=1,B=2,C=3**

**We can solve this as**

**1 2 + 3 - 2 1 + 3 pow -**

**At first 1 and 2 are pushed in a stack then operator is scanned as the rule of evaluation of postfix 2 operand before operator is poped and operation is performed and pushed into stack then expression becomes as**

**3 3 - 2 1 + 3 pow -**

**Similarly 3 and 3 are pushed in stack then - is scanned and 3 & 3 are poped and operation is performed and again pushed back to stack**

**0 2 1 + 3 pow-**

**Similarly 2 and 1 is poped after scanning + operator and operation is done**

**0 3 3 pow -**

**Now power is scanned the 3,3 is poped and operation is performed and stored in stack**

**0 27-**

**Now 27 and 0 is poped after - is scanned and operation is done i.e**

**-27**

**Hence -27 is the final answer**

**2. ABC+\*CBA-+\***

Here the given postfix expression is

**ABC+\*CBA-+\***

And the value is **A=1,B=2,C=3**

**We can solve this as**

**1 2 3 + \* 3 2 1- + \***

**At first 1 2 and 3 are pushed in a stack then operator is scanned as the rule of evaluation of postfix 2 operand before operator is poped and operation is performed and pushed into stack then expression becomes as**

**1 5 \* 3 2 1 - + \***

**Similarly 1 and 5 are pushed in stack then \* is scanned and 1 & 5 are poped and operation is performed and again pushed back to stack**

**5 3 2 1 -+\***

**Similarly 5 3 2 1 are pushed in stack then - is scanned and 1 & 2 are poped and operation is performed and again pushed back to stack**

**5 3 1 +\***

**Now + is scanned then 1,3 is poped and operation is performed and stored in stack**

**5 4\***

**Now \* is scanned then 4,5 is poped and operation is performed and stored in stack**

**20**

**Hence 20 is final answer**

##### **Create a table and convert the following infix to postfix**

**1. A - B + C becomes A B - C +**

|  |  |  |
| --- | --- | --- |
| **Current symbol** | **opstack** | **poststack** |
| **A** |  | **A** |
| **-** | **-** | **A** |
| **B** | **-** | **AB** |
| **+** | **+** | **AB-** |
| **c** | **+** | **AB-C+** |

**2. A \* B ^ C + D becomes A B C ^ \* D +**

|  |  |  |
| --- | --- | --- |
| **Current symbol** | **opstack** | **poststack** |
| **A** |  | **A** |
| **\*** | **\*** | **A** |
| **B** | **\*** | **AB** |
| **^** | **\*^** | **AB** |
| **C** | **\*^** | **ABC** |
| **+** | **\*^** | **ABC^** |
|  | **\*** | **ABC^\*** |
|  | **+** | **ABC^\*** |
| **D** | **+** | **ABCD^\*D** |
|  |  | **ABCD^\*D+** |

**3. A \* (B + C \* D) + E becomes A B C D \* + \* E +**

|  |  |  |
| --- | --- | --- |
| **Current symbol** | **opstack** | **poststack** |
| **A** |  | **A** |
| **\*** | **\*** | **A** |
| **(** | **\*(** | **A** |
| **B** | **\*(** | **AB** |
| **+** | **\*(+** | **AB** |
| **C** | **\*(+** | **ABC** |
| **\*** | **\*(+\*** | **ABC** |
| **D** | **\*(+\*** | **ABCD** |
| **)** | **\*** | **ABCD\*+** |
|  |  | **ABCD\*+\*** |
| **+** | **+** | **ABCD\*+\*** |
| **E** | **+** | **ABCD\*+\*E** |
|  |  | **ABCD\*+\*E+** |