

**“Assignment”**

|  |  |
| --- | --- |
| **Course Code:** | **Course Title:** |
| CSE 134 | Data Structure |

|  |  |
| --- | --- |
| **Assignment Details** | |
| Assignment Topic | : Prefix to Infix and Infix to Prefix Conversion using Stack |
| Submission Date | : 01/10/2023 |

|  |  |
| --- | --- |
| **Submitted To:** | **Submitted By:** |
| **Amatul Bushra Akhi**  Assistant Professor  Department of CSE.  Daffodil International University | * Munna Biswas (221-15-**5261**) * Naimul Islam (221-15-**4941**) * Riad Hossain (221-15-**5249**) * Ahsanul Hoque (221-15-**5330**) * Monowar Afsar (221-15-**5366**)   Section: 61-J  Department of CSE.  Daffodil International University |

**Expression Conversion (prefix >< infix)**

# **What is prefix notation?**

When the operator is written in before the operands, then it is known as infix notation.

Example: +AB, \*XY, -56

# **What is infix notation?**

When the operator is written in between the operands, then it is known as infix notation.

Example: A+B, X\*Y, 5-6

There are six types of conversion possible for three types of expression. In this document we will solve only two of them.

1. Prefix to Infix Conversion
2. Infix to Prefix Conversion

**Prefix to Infix Conversion:**

**Steps:**

1. Read the Prefix expression in reverse order (from right to left)
2. If the symbol is an operand, then push it onto the Stack.
3. If the symbol is an operator, then pop two operands from the Stack and create it a string by placing the operator in between the operands an push it into the stack.
4. Repeat the above steps until the end of Prefix expression.

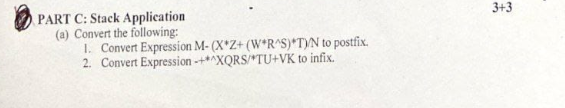
**Infix to Prefix Conversion:**

**Steps:**

1. Scan the infix expression right to left.
2. If the symbol is an operand, then push into prefix expression.
3. If the symbol is an operator, then push it into the stack.
4. Highest priority operator will not stay (will be placed in the prefix expression) in stack when lowest priority operator will inserted.
5. If the operator is ')', then push it into the stack.
6. If the operator is '(', then pop all the operators from the stack till it finds ) opening bracket in the stack.
7. Repeat the above steps until the end of infix expression.
8. At the end, reverse the output.

**Solutions of previous years questions**

**Spring 2022**

****

Prefix expression: -+\*^XQRS/\*TU+VK

Infix conversion-

**Stack**

**Symbol**

K

K V

(V+K)

(V+K) U

(V+K) U T

(V+K) (T\*U)

((T\*U) / (V+K))

((T\*U) / (V+K)) S

((T\*U) / (V+K)) S R

((T\*U) / (V+K)) S R Q

((T\*U) / (V+K)) S R Q X

((T\*U) / (V+K)) S R (X^Q)

((T\*U) / (V+K)) S ((X^Q) \* R)

((T\*U) / (V+K)) (((X^Q) \* R) +S)

(((X^Q) \* R) +S) - ((T\*U) / (V+K))

K

V

+

U

T

\*

/

S

R

Q

X

^

\*

+

-

Infix Expression: (((X^Q) \* R) +S) - ((T\*U) / (V+K))

**Again** **Infix to prefix->**

Symbol Stack prefix expression (reverse)

) )

) ))

K )) K

+ ))+ K

V ))+ KV

( ) KV+

/ )/ KV+

) )/) KV+

U )/) KV+U

\* )/)\* KV+U

T )/)\* KV+UT

( )/ KV+UT\*

( KV+UT\*/

- - KV+UT\*/

) -) KV+UT\*/

S -) KV+UT\*/S

+ -)+ KV+UT\*/S

) -)+) KV+UT\*/S

R -)+) KV+UT\*/SR

\* -)+)\* KV+UT\*/SR

) -)+)\*) KV+UT\*/SR

Q -)+)\*) KV+UT\*/SRQ

^ -)+)\*)^ KV+UT\*/SRQ

X -)+)\*)^ KV+UT\*/SRQX

( -)+)\* KV+UT\*/SRQX^

( -)+ KV+UT\*/SRQX^\*

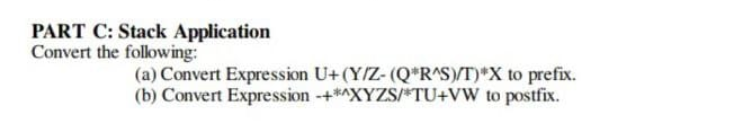
( - KV+UT\*/SRQX^\*+

KV+UT\*/SRQX^\*+-

After reversing -

Prefix Expression: -+\*^XQRS/\*TU+VK

**Spring 2021**

****

Infix Expression: U+(Y/Z-(Q\*R^S)/T)\*X

Prefix Conversion-

Symbol Stack prefix expression (reverse)

X X

\* \* X

) \* ) X

T \* ) XT

/ \* ) / XT

) \* ) / ) XT

S \* ) / ) XTS

^ \* ) / ) ^ XTS

R \* ) / ) ^ XTSR

\* \* ) / ) \* XTSR^

Q \* ) / ) \* XTSR^Q

( \* ) / XTSR^Q\*

- \* ) - XTSR^Q\*/

Z \* ) - XTSR^Q\*/Z

/ \* ) -/ XTSR^Q\*/Z

Y \* ) -/ XTSR^Q\*/ZY

( \* XTSR^Q\*/ZY/-

+ + XTSR^Q\*/ZY/-\*

U XTSR^Q\*/ZY/-\*U+

After reversing -

Prefix Expression: +U\*-/YZ/\*Q^RSTX

Again prefix to infix->

**Symbol Stack**

X

X

XT

XTS

XTSR

XT(R^S)

XT(R^S)Q

XT(Q\*(R^S))

X((Q\*(R^S))/T)

X((Q\*(R^S))/T)Z

X((Q\*(R^S))/T)ZY

X((Q\*(R^S))/T) (Y/Z)

T

S

R

^

Q

\*

/

Z

Y

/

-

X ((Y/Z) - ((Q\*(R^S))/T))

X ((Y/Z) - ((Q\*(R^S))/T)) U

X (U+((Y/Z) - ((Q\*(R^S))/T)))

(U+((Y/Z) - ((Q\*(R^S))/T))) \*X

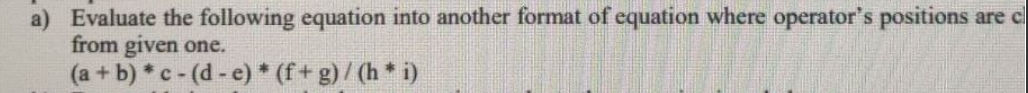
U

+

\*

Infix expression: (U+((Y/Z) - ((Q\*(R^S))/T))) \*X

**Fall 2020**



Infix Expression: (a + b) \* c - (d - e) \* (f + g) / (h \* i)

Symbol Stack prefix expression (rev)

) )

i ) i

\* )\* i

h )\* ih

( ih\*

/ / ih\*

) /) ih\*

g /) ih\*g

+ /)+ ih\*g

f /)+ ih\*gf

( / ih\*gf+

\* /\* ih\*gf+

) /\*) ih\*gf+

e /\*) ih\*gf+e

- /\*)- ih\*gf+e

d /\*)- ih\*gf+ed

( /\* ih\*gf+ed-

- - ih\*gf+ed-\*/

c - ih\*gf+ed-\*/c

\* -\* ih\*gf+ed-\*/c

) -\*) ih\*gf+ed-\*/c

b -\*) ih\*gf+ed-\*/cb

+ -\*)+ ih\*gf+ed-\*/cb

a -\*)+ ih\*gf+ed-\*/cba

( -\* ih\*gf+ed-\*/cba+

ih\*gf+ed-\*/cba+\*-

After reversing -

Prefix Expression: **- \* + a b c / \* - d e + f g \* h i**

**Prefix to infix->**

**Symbol Stack**

i

ih

(h\*i)

(h\*i) g

(h\*i) gf

(h \* i) (f + g)

(h \* i) (f + g) e

(h \* i) (f + g) e d

(h \* i) (f + g) (d – e)

i

h

\*

g

f

+

e

d

-

\*

(h \* i) ((d – e) \* (f + g))

(((d – e) \* (f + g)) / (h \* i))

(((d – e) \* (f + g)) / (h \* i)) c

(((d – e) \* (f + g)) / (h \* i)) cb

(((d – e) \* (f + g)) / (h \* i)) cba

(((d – e) \* (f + g)) / (h \* i)) c (a + b)

(((d – e) \* (f + g)) / (h \* i)) ((a + b) \* c)

((a + b) \* c) -(((d – e) \* (f + g)) / (h \* i))

/

c

b

a

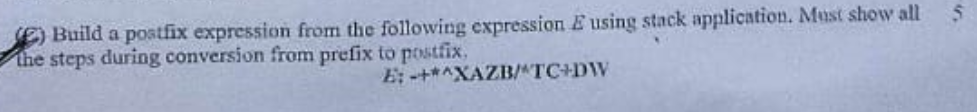
+

\*

-

Infix Expression: ((a + b) \* c) -(((d – e) \* (f + g)) / (h \* i))

**Spring 2023**

****

Prefix Expression: -+\*^XAZB/\*TC+DW

Infix conversion:

Symbol Stack

W

W

WD

(D + W)

(D+W) C

(D+W) CT

(D+W) (T\*C)

((T\*C) / (D+W))

((T\*C) / (D+W)) B

((T\*C) / (D+W)) BZ

((T\*C) / (D+W)) BZA

((T\*C) / (D+W)) BZAX

((T\*C) / (D+W)) BZ (X^A)

((T\*C) / (D+W)) B ((X^A) \* Z)

((T\*C) / (D+W)) (((X^A) \* Z) + B)

((((X^A) \* Z) + B) - ((T\*C) / (D+W)))

D

+

C

T

\*

/

B

Z

A

X

^

\*

+

-

Infix Expression: ((((X^A) \* Z) + B) - ((T\*C) / (D+W)))

Again infix to prefix:

Symbol Stack prefix expression(rev)

) )

) ))

) )))

W ))) W

+ )))+ W

D )))+ WD

( )) WD+

/ ))/ WD+

) ))/) WD+

C ))/) WD+C

\* ))/)\* WD+C

T ))/)\* WD+CT

( ))/ WD+CT\*

( ) WD+CT\*/

- )- WD+CT\*/

) )-) WD+CT\*/

B )-) WD+CT\*/B

+ )-)+ WD+CT\*/B

) )-)+) WD+CT\*/B

Z )-)+) WD+CT\*/BZ

\* )-)+)\* WD+CT\*/BZ

) )-)+)\*) WD+CT\*/BZ

A )-)+)\*) WD+CT\*/BZA

^ )-)+)\*)^ WD+CT\*/BZA

X )-)+)\*)^ WD+CT\*/BZAX

( )-)+)\* WD+CT\*/BZAX^

( )-)+ WD+CT\*/BZAX^\*

( )- WD+CT\*/BZAX^\*+

( WD+CT\*/BZAX^\*+-

After reversing:

Prefix Expression: -+\*^XAZB/\*TC+DW