Data structure and algorithm
hw04
part01

mohammad ashraf yawar
161044123

I)

a) infix to postfix:

first we convert the infix to postfix and then evaluate ans demonstrate it:

A + ((B – C * D) / E) + F – G / H

A + ((B –(C D *) / E) + F – G / H

A + (B C D * -) / E) + F – G / H

A + (B C D * - E /) + F – G / H

(A B C D * - E / +) + F – G / H

(A B C D * - E / + F +) – G / H

(A B C D * - E / + F +) – (G H /)

(A B C D * - E / + F + G H / -)

Evaluation:

so evaluate such expression we have the bellow algorithm :

ALGORITHM:

0. create an empty stack of integers
1. start from the end of tokenArray toward first element in the array
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

let's give some values to the variables:

double A = 1 , B = 2 , C = 4 , D = 6 , E = 2 , F = 1, G = 1 , H = 4

step 1)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

myStack

1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 2)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

myStack

2
1

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 3)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

myStack

4
2
1

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 4)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

myStack

6
4
2
1

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 5)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

4 * 6 = 24

myStack

24
2
1

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 6)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

2 - 24 = -22

myStack

-22
1

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 7)

tokenArray with length of 15

1	**2**	**4**	**6**	*****	**-**	**2**	**/**	**+**	**1**	**+**	**1**	**4**	**/**	**-**

myStack

2
-22
1

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 8)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

-22 / 2 = -11

myStack

-11
1

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 9)

tokenArray with length of 15

1	**2**	**4**	**6**	*****	**-**	**2**	**/**	**+**	**1**	**+**	**1**	**4**	**/**	**-**

1 – 11 =-10

myStack

-10

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 10)

tokenArray with length of 15

1	**2**	**4**	**6**	*****	**-**	**2**	**/**	**+**	**1**	**+**	**1**	**4**	**/**	**-**

myStack

1
-10

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 11)

tokenArray with length of 15

1	**2**	**4**	**6**	*****	**-**	**2**	**/**	**+**	**1**	**+**	**1**	**4**	**/**	**-**

1 – 10 =-9

myStack

-9

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 12)

tokenArray with length of 15

1	**2**	**4**	**6**	*****	**-**	**2**	**/**	**+**	**1**	**+**	**1**	**4**	**/**	**-**

myStack

1
-9

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 13)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

myStack

4
1
-9

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 14)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

1 / 4 =0.25

myStack

0.25
-9

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 15)

tokenArray with length of 15

1	2	4	6	*	-	2	/	+	1	+	1	4	/	-

-9 – 0.25 = -9.25

myStack

-9.25

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 16)

tokenArray with length of 15

1	**2**	**4**	**6**	*****	**-**	**2**	**/**	**+**	**1**	**+**	**1**	**4**	**/**	**-**

myStack

-9.25

1. create an empty stack of integers
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

return -9.25 as result

b) infix to prefix:

first we convert the infix to prefix and then evaluate ans demonstrate it:

A + ((B – C * D) / E) + F – G / H

A + ((B – (* C D) / E) + F – G / H

A + ((- B * C D) / E) + F – G / H

A + (/ - B * C D E) + F – G / H

(+ A / - B * C D E) + F – G / H

(+ A / - B * C D E) + F – (/ G H)

(+ + A / - B * C D E F) – (/ G H)

- + + A / - B * C D E F / G H

Evaluation:

so evaluate such expression we have the bellow algorithm :
this time instead of starting from the beginning of the tokenArray we start from tail to first element.

ALGORITHM:

1. create an empty stack of integers
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

let's give some values to the variables:

double A = 1 , B = 2 , C = 4 , D = 6 , E = 2 , F = 1, G = 1 , H = 4

step 1)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

4

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 2)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

1
4

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 3)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

¼ = 0.25

myStack

0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 4)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 5)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

2
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 6)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

6
2
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 7)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

4
6
2
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 8)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

4 * 6 = 24

myStack

24
2
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 9)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

2
24
2
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 10)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

2 - 24 = -22

myStack

-22
2
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 11)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

-22 /1 = -11

myStack

-11
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 12)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

1
-11
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 13)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1 - 11 = -10

myStack

-10
1
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 14)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

-10 +1 = -9

myStack

-9
0.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 15)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

-9 -0.25 = -9.25

myStack

-9.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 16)

tokenArray with length of 15

-	+	+	1	/	-	2	*	4	6	2	1	/	1	4
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

-9.25

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

returns -9.25 as resutl

II)

a) infix to postfix:

! (A && ! ((B < C) || (C>D))) || (C < E)

! (A && ! ((B C <) || (C D >))) || (C E <)

! (A && ! (B C < C D > ||)) || (C E <)

! (A && B C < C D > || !) || (C E <)

! (A B C < C D > || ! &&) || (C E <)

(A B C < C D > || ! && !) || (C E <)

A B C < C D > || ! && ! C E < ||

Evaluation:

so evaluate such expression we have the bellow algorithm :

ALGORITHM:

1. create an empty stack of integers
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

let's give some values to the variables:

int A = 1 , B = 2 , C = 4 , D = 6 , E = 2

step 1)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 2)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

2
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 3)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

4
2
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 4)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

2 < 4 = true = 1

myStack

1
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 5)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

4
1
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 6)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

6
4
1
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 7)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

4 > 6 = false = 0

myStack

0
1
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 8)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1 || 0 = true = 1

myStack

1
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 9)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

!1 = false = 0

myStack

0
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 10)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1 && 0 = false = 0

myStack

0

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 11)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

!0 = true = 1

myStack

1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 12)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

4
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 13)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

2
4
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 14)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

2 < 4 = true = 1

myStack

1
1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 15)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1 || 1 = true = 1

myStack

1

1. create an empty stack of integers (myStack)
2. while there are more tokens
3. get the next token
4. if the first character of the token is a digit
5. push the token on the stack
6. else if the token is an operator
7. pop the right operand off the stack
8. pop the left operand off the stack
9. evaluate the operation
10. push the result onto the stack
11. pop the stack and return the result

step 16)

tokenArray with length of 15

1	2	4	<	4	6	>	||	!	&&	!	4	2	<	||
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

1

```
1. create an empty stack of integers     (myStack)
2. while there are more tokens
3.      get the next token
4.      if the first character of the token is a digit
5.              push the token on the stack
6.      else if the token is an operator
7.              pop the right operand off the stack
8.              pop the left operand off the stack
9.              evaluate the operation
10.             push the result onto the stack
11. pop the stack and return the result
```

retuns 1 as true

b) infix to prefix:

! (A && ! ((B < C) || (C>D))) || (C < E)

! (A && ! ((< B C) || (>C D))) || (< C E)

! (A && ! (|| < B C > C D)) || (< C E)

! (A && (!|| < B C > C D)) || (< C E)

! (&& A! || < B C >C D) || (< C E)

(! && A! || < B C > C D) || (< C E)

|| ! && A! || < B C > C D < C E

Evaluation:

so evaluate such expression we have the bellow algorithm :
this time instead of starting from the beginning of the tokenArray we start from tail to first element.

ALGORITHM:

1. create an empty stack of integers
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

let's give some values to the variables:

double A = 1 , B = 2 , C = 4 , D = 6 , E = 2

step 1)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

2

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 2)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

4
2

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 3)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

4 < 2 = false = 0

myStack

0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 4)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

6
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 5)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

4
6
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 6)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

4 < 6 = true = 1

myStack

1
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 7)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

4
1
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 8)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

2
4
1
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 9)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

2 < 4 = true = 1

myStack

1
1
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 10)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1 || 1 = true = 1

myStack

1
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 11)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

! 1 = false = 0

myStack

0
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 12)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

1
0
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 13)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1 && 0 = false = 0

myStack

0
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 14)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

!0 = true = 1

myStack

1
0

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 15)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1 || 0 = true = 1

myStack

1

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

step 16)

tokenArray with length of 15

||	!	&&	1	!	||	<	2	4	>	4	6	<	4	2
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

myStack

1

1. create an empty stack of integers (myStack)
2. start from the end of tokenArray toward first element in the array
3. while there are more tokens
4. get the next token
5. if the first character of the token is a digit
6. push the token on the stack
7. else if the token is an operator
8. pop the right operand off the stack
9. pop the left operand off the stack
10. evaluate the operation
11. push the result onto the stack
12. pop the stack and return the result

returns true as result