GIT

Department of Computer Engineering

CSE 222/505 - Spring 2020

Homework 4 – Report

Elif Goral

171044003

Q1)

i)

Infix expression: A + ((B – C * D) / E) + F – G / H

Input expression	Operator Stack	Postfix expression
A	empty	A
+	+ (push + to the stack)	A
(	+ ((push (to the stack)	A
(	+ (((push (to the stack)	A
B	+ ((	A B
-	+ ((- (push – to the stack)	A B
C	+ ((-	A B C
*	+ ((- * (push * to the stack)	A B C
D	+ ((- *	A B C D
)	+ ((pop * from the stack) (pop – from the stack)	A B C D * -

	(pop (from the stack)	
/	+ (/ *(push / to the stack)*	A B C D * -
E	+ (/	A B C D * - E
)	+ *(pop / from the stack)* *(pop (from the stack)*	A B C D * - E /
+	+ *(pop+ from the stack)* *(push + to the stack)*	A B C D * - E / +
F	+	A B C D * - E / + F
-	- *(pop + from the stack)* *(push – to the stack)*	A B C D * - E / + F +
G	-	A B C D * - E / + F + G
/	- / *(push / to the stack)*	A B C D * - E / + F + G
H	- /	A B C D * - E / + F + GH

Pop / from the stack

Pop – from the stack

Postfix expression: A B C D * - E / + F + GH / -

Infix expression: A + ((B – C * D) / E) + F – G / H

Step1 : reverse the infix expression

Reverse: H / G – F (E / (D * C – B)) + A

Step2: convert reverse expression to postfix expression

Input expression	*Operator Stack*	*Postfix expression*
H	empty	H
/	/ (push / to the stack)	H
G	/	H G
-	- (pop / from the stack) (push – to the stack)	H G /
F	-	H G / F
+	+ (pop – from the stack) (push + to the stack)	H G / F -
(	+ ((push (to the stack)	H G / F -

E	*+ (*	*H G / F – E*
/	*+ (/* *(push / to the stack)*	*H G / F – E*
(	*+ (/ (* *(push (to the stack)*	*H G / F – E*
D	*+ (/ (*	*H G / F – E D*
***	*+ (/ (** *(push * to the stack)*	*H G / F – E D*
C	*+ (/ (**	*H G / F – E D C*
-	*+ (/ (-* *(pop * from the stack)* *(push – to the stack)*	*H G / F – E D C **
B	*+ (/ (-*	*H G / F – E D C * B*
)	*+ (/* *(pop - from the stack)* *(pop (from the stack)*	*H G / F – E D C * B -*
)	*+* *(pop / from the stack)* *(pop (from the stack)*	*H G / F – E D C * B - /*
+	*+* *(pop + from the stack)* *(push + to the stack)*	*H G / F – E D C * B - / +*
A	*+*	*H G / F – E D C * B - / + A*

Pop + from the stack

*Result of step2: H G / F – E D C * B - / + A +*

Step3: reverse the result of step2.

*Prefix expression: + A + / - B * C D E – F / G H*

*İnfix: A + ((B – C * D) / E) + F – G / H*

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

*Postfix: A B C D * - E / + F + GH / -*

For example;

A = 5, B = 30, C = 3, D = 9 , E = 3 , F = 8, G =16 , H =4

İnfix evaluation:

*= 5 + ((30 – 3 * 9) / 3) + 8 – 16 / 4*

= 5 + ((30 – 27) / 3) + 8 – 16 / 4

= 5 + (3 / 3) + 8 – 16 / 4

= 5 + 1 + 8 – 4

= 10

Prefix evaluation:

*= + 5 + / - 30 * 3 9 3 – 8 / 16 4*

*= + 5 + / - 30 * 3 9 3 – 8 4*

*= + 5 + / - 30 * 3 9 3 4*

= + 5 + / - 30 27 3 4

= + 5 + / 3 3 4

= + 5 + 1 4

= + 5 5

= 10

Postfix Evaluation:

= 5 30 3 9 * - 3 / + 8 + 16 4 / -

= 5 30 27 – 3 / + 8 + 16 4 / -

= 5 3 3 / + 8 + 16 4 / -

= 5 1 + 8 + 16 4 / -

= 6 8 + 16 4 / -

= 14 16 4 / -

= 14 4 –

= 10

Result:

Prefix evaluation = postfix evaluation = infix evalutaion= 10

ii)

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

Input expression	*Operator Stack*	*Postfix expression*
!	*!* *(push ! to the stack)*	
(	*! (* *(push (to the stack)*	
A	*! (*	*A*
&&	*! (&&* *(push && to the stack)*	*A*
!	*! (&& !* *(push ! to the stack)*	*A*
(	*! (&& ! (* *(push (to the stack)*	*A*
(	*! (&& ! ((* *(push (to the stack)*	*A*
B	*! (&& ! ((*	*A B*
<	*! (&& ! ((<*	*A B*
C	*! (&& ! ((<*	*A B C*
)	*! (&& ! (* *(pop < from the stack)* *(pop (from the stack)*	*A B C <*
||	*! (&& ! (||* *(push || to the stack)*	*A B C <*
(	*! (&& ! (|| (* *(push (to the stack)*	*A B C <*

C	*! (&& ! (|| (*	*A B C < C*
>	*! (&& ! (|| (>* *(push > to the stack)*	*A B C < C*
D	*! (&& ! (|| (>*	*A B C < C D*
)	*! (&& ! (||* *(pop > from the stack)* *(pop (from the stack)*	*A B C < C D >*
)	*! (&& !* *(pop || from the stack)* *(pop (from the stack)*	*A B C < C D > ||*
)	*!* *(pop ! from the stack)* *(pop && from the stack)* *(pop (from the stack)*	*A B C < C D > || ! &&*
||	*||* *(pop ! from the stack)* *(push || to the stack)*	*A B C < C D > || ! && !*
(	*|| (* *(push (to the stack)*	*A B C < C D > || ! && !*
C	*|| (*	*A B C < C D > || ! && ! C*
<	*|| (<* *(push < to the stack)*	*A B C < C D > || ! && ! C*
E	*|| (<*	*A B C < C D > || ! && ! C E*
)	*||* *(pop < from the stack)*	*A B C < C D > || ! && ! C E <*

	(pop (from the stack)	

Poll || from the stack

Postfix expression: *A B C < C D > || ! && ! C E < ||*

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

Step1 : reverse the infix expression

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

Step2: convert reverse expression to postfix expression

Input expression	*Operator Stack*	*Postfix expression*
(	*(* *(push (to the stack)*	
E	*(*	*E*
<	*(<* *(push < to the stack)*	*E*
C	*(<*	*E C*
)	*Empty* *(pop < from the stack)* *(pop (from the stack)*	*E C <*

||	*||* *(push || to the stack)*	*E C <*
(	*|| (* *(push (to the stack)*	*E C <*
(	*|| ((* *(push (to the stack)*	*E C <*
(	*|| (((* *(push (to the stack)*	*E C <*
D	*|| (((*	*E C < D*
>	*|| (((>* *(push > to the stack)*	*E C < D*
C	*|| (((>*	*E C < D C*
)	*|| ((* *(pop > from the stack)* *(pop (from the stack)*	*E C < D C >*
||	*|| ((||* *(push || to the stack)*	*E C < D C >*
(	*|| ((|| (* *(push (to the stack)*	*E C < D C >*
C	*|| ((|| (*	*E C < D C > C*
<	*|| ((|| (<* *(push < to the stack)*	*E C < D C > C*
B	*|| ((|| (<*	*E C < D C > C B*
)	*|| ((||* *(pop < from the stack)*	*E C < D C > C B <*

	(pop (from the stack)	
)	*|| (* *(pop || from the stack)* *(pop (from the stack)*	*E C < D C > C B < ||*
!	*|| (!* *(push ! to the stack)*	*E C < D C > C B < ||*
&&	*|| (&&* *(pop ! from the stack)* *(push && to the stack)*	*E C < D C > C B < || !*
A	*|| (&&*	*E C < D C > C B < || ! A*
)	*||* *(pop && from the stack)* *(pop (from the stack)*	*E C < D C > C B < || ! A &&*
!	*|| !* *(push ! to the stack)*	*E C < D C > C B < || ! A &&*

Pop ! from the stack

Pop || from the stack

Result of step2: E C < D C > C B < || ! A && ! ||

Step3: reverse the result of step2.

Prefix expression : || ! && A ! || < B C > C D < C E

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

Prefix exp: *|| ! && A ! || < B C > C D < C E*

Postfix exp: *A B C < C D > || ! && ! C E < ||*

For example: *A = 7 , B = 8 , C = 10, D = 5, E = 16*

Infix evalutaion:

= ! (7 && ! ((8 < 10) || (10 > 5))) || (10 < 16)

= ! (7 && ! (true || true)) || true

= ! (7 && ! (true)) || true

= ! (7 && (false)) || true

= ! (false) || true

= true || true

= true

Prefix evaluation:

= || ! && 7 ! || < 8 10 > 10 5 < 10 16

= || ! && 7 ! || < 8 10 > 10 5 (true)

= || ! && 7 ! || < 8 10 (true) (true)

= || ! && 7 ! || (true) (true) (true)

= || ! && 7 ! (true) (true)

= || ! && 7 (false) (true)

= || ! (false) (true)

= || (true) (true)

= true

Postfix evaluation:

= 7 8 10 < 10 5 > || ! && ! 10 16 < ||

= 7 (true) 10 5 > || ! && ! 10 16 < ||

= 7 (true) (true) || ! && ! 10 16 < ||

= 7 (true) ! && ! 10 16 < ||

= 7 (false) && ! 10 16 < ||

= (false) ! 10 16 < ||

= (true) 10 16 < ||

= (true) (true) ||

= true

Result:

Infix evaluation = prefix evaluation = postfix evaluation = true

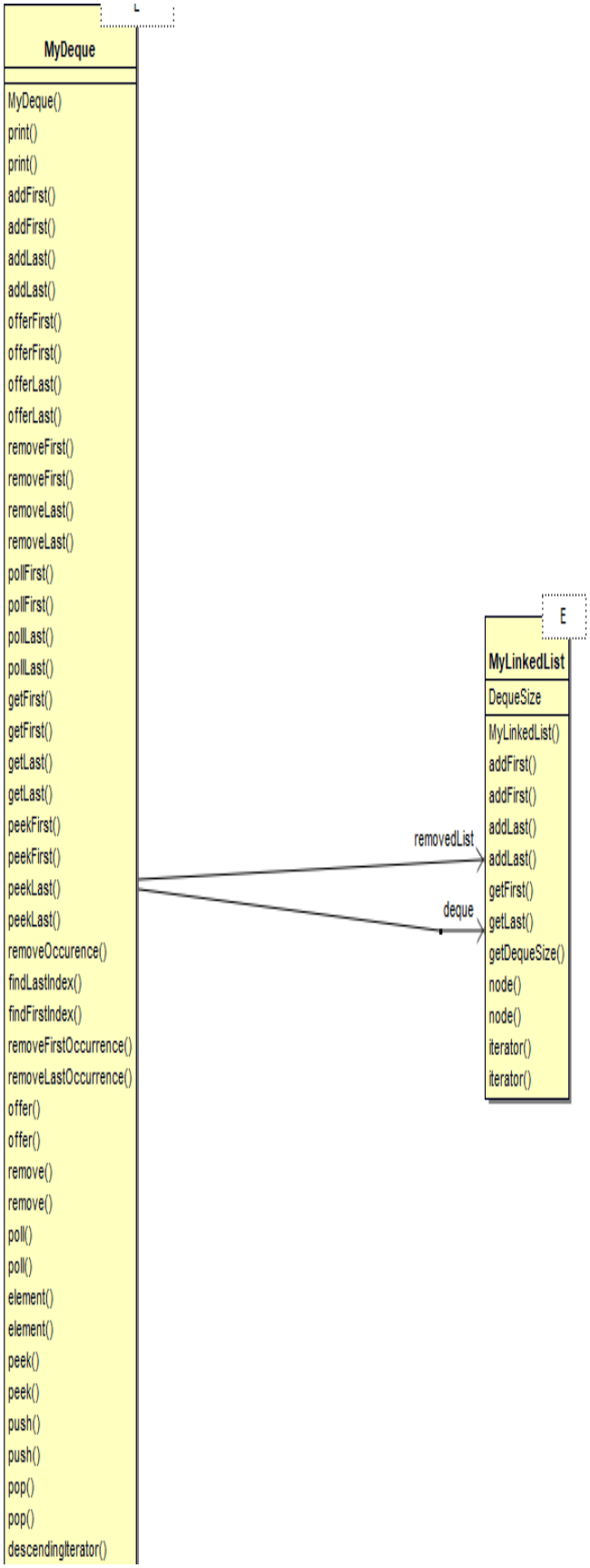
Q2)

Problem Solutions Approach

In this Project, I create my deque class which implements Deque interface and extends AbstractCollection and my linked list classes. My linked list class has iterator and node classes which are created by me.

In deque class, as a first, I create a two linked list which is constitued by my linked list class. First linked list is for my deque, second linked list is for removed nodes. I use a removed node, if any available, when a new node is needed instead of creating a new node. With this , I save time for constructing new nodes and garbage collection the nodes that are not used anymore. When I overriding a methods, I write helper methods either for chosing the list type. For example , add method avaliable for both removedList and deque. I can not add a parameter to override method,So I create an other method and send list type to method. Most of the methods, I used that strategy. In linked list class I have add methods to,to move in nodes.

Diagram


MyDeque
MyDeque()
print()
print()
addFirst()
addFirst()
addLast()
addLast()
offerFirst()
offerFirst()
offerLast()
offerLast()
removeFirst()
removeFirst()
removeLast()
removeLast()
pollFirst()
pollFirst()
pollLast()
pollLast()
getFirst()
getFirst()
getLast()
getLast()
peekFirst()
peekFirst()
peekLast()
peekLast()
removeOccurence()
findLastIndex()
findFirstIndex()
removeFirstOccurrence()
removeLastOccurrence()
offer()
offer()
remove()
remove()
poll()
poll()
element()
element()
peek()
peek()
push()
push()
pop()
pop()
descendingIterator()
removedList
deque
E
MyLinkedList
DequeSize
MyLinkedList()
addFirst()
addFirst()
addLast()
addLast()
getFirst()
getLast()
getDequeSize()
node()
node()
iterator()
iterator()

Test Cases and Results

```
addFirst 12 :
printing deque...
dequesize: 1
12
printing removed list...
list is empty
getFirst:12
peekFirst: 12
getLast:12
peekLast: 12
-----------------------

addLast 12
printing deque...
dequesize: 2
12 12
printing removed list...
list is empty
getFirst:12
peekFirst: 12
getLast:12
peekLast: 12
-----------------------

addFirst 123 :
printing deque...
dequesize: 3
123 12 12
printing removed list...
list is empty
getFirst:123
peekFirst: 123
getLast:12
peekLast: 12
-----------------------

addLast 45 :
printing deque...
dequesize: 4
123 12 12 45
printing removed list...
list is empty
getFirst:123
peekFirst: 123
getLast:45
peekLast: 45
-----------------------
```

```
-----------------------

offerFirst 142 :
printing deque...
dequesize: 5
142 123 12 12 45
printing removed list...
list is empty
getFirst:142
peekFirst: 142
getLast:45
peekLast: 45
-----------------------

offerLast 723 :
printing deque...
dequesize: 6
142 123 12 12 45 723
printing removed list...
list is empty
getFirst:142
peekFirst: 142
getLast:723
peekLast: 723
-----------------------

addLast 4456 :
printing deque...
dequesize: 7
142 123 12 12 45 723 4456
printing removed list...
list is empty
getFirst:142
peekFirst: 142
getLast:4456
peekLast: 4456
-----------------------

removeFirst:
printing deque...
dequesize: 6
123 12 12 45 723 4456
printing removed list...
142
getFirst:123
peekFirst: 123
getLast:4456
peekLast: 4456
-----------------------
```

```
addFirst 78:
printing deque...
dequesize: 7
78 123 12 12 45 723 4456
printing removed list...
list is empty
getFirst:78
peekFirst: 78
getLast:4456
peekLast: 4456
-----------------------

pollFirst:
printing deque...
dequesize: 6
123 12 12 45 723 4456
printing removed list...
78
getFirst:123
peekFirst: 123
getLast:4456
peekLast: 4456
-----------------------

addFirst 70 :
printing deque...
dequesize: 7
70 123 12 12 45 723 4456
printing removed list...
list is empty
getFirst:70
peekFirst: 70
getLast:4456
peekLast: 4456
-----------------------

addLast 169 :
printing deque...
dequesize: 8
70 123 12 12 45 723 4456 169
printing removed list...
list is empty
getFirst:70
peekFirst: 70
getLast:169
peekLast: 169
-----------------------
```

```
removeFirst
printing deque...
dequesize: 7
123 12 12 45 723 4456 169
printing removed list...
70
getFirst:123
peekFirst: 123
getLast:169
peekLast: 169
-----------------------

removeFirst:
printing deque...
dequesize: 6
12 12 45 723 4456 169
printing removed list...
123 70
getFirst:12
peekFirst: 12
getLast:169
peekLast: 169
-----------------------

Removefirst:
printing deque...
dequesize: 5
12 45 723 4456 169
printing removed list...
12 123 70
getFirst:12
peekFirst: 12
getLast:169
peekLast: 169
-----------------------

addLast 463
printing deque...
dequesize: 6
12 45 723 4456 169 463
printing removed list...
123 70
getFirst:12
peekFirst: 12
getLast:463
peekLast: 463
-----------------------
```

```
addfirst 178
printing deque...
dequesize: 7
178 12 45 723 4456 169 463
printing removed list...
70
getFirst:178
peekFirst: 178
getLast:463
peekLast: 463
-----------------------

removeFirst
printing deque...
dequesize: 6
12 45 723 4456 169 463
printing removed list...
178 70
getFirst:12
peekFirst: 12
getLast:463
peekLast: 463
-----------------------

removeFirst
printing deque...
dequesize: 5
45 723 4456 169 463
printing removed list...
12 178 70
getFirst:45
peekFirst: 45
getLast:463
peekLast: 463
-----------------------

addLast 169
printing deque...
dequesize: 6
45 723 4456 169 463 169
printing removed list...
178 70
getFirst:45
peekFirst: 45
getLast:169
peekLast: 169
-----------------------
```

```
removeFirstOccurence 723
printing deque...
dequesize: 5
45 4456 169 463 169
printing removed list...
723 178 70
getFirst:45
peekFirst: 45
getLast:169
peekLast: 169
-----------------------

addLast 1
printing deque...
dequesize: 6
45 4456 169 463 169 1
printing removed list...
178 70
getFirst:45
peekFirst: 45
getLast:1
peekLast: 1
-----------------------

addLast 169
printing deque...
dequesize: 7
45 4456 169 463 169 1 169
printing removed list...
70
getFirst:45
peekFirst: 45
getLast:169
peekLast: 169
-----------------------


offer 67
printing deque...
dequesize: 8
45 4456 169 463 169 1 169 67
printing removed list...
list is empty
getFirst:45
peekFirst: 45
getLast:67
peekLast: 67
-----------------------

element: 45

-----------------------
peek: 45

-----------------------
```

```
----------------------
push 100
printing deque...
dequesize: 9
100 45 4456 169 463 169 1 169 67
printing removed list...
list is empty
getFirst:100
peekFirst: 100
getLast:67
peekLast: 67
----------------------

poll
printing deque...
dequesize: 8
45 4456 169 463 169 1 169 67
printing removed list...
100
getFirst:45
peekFirst: 45
getLast:67
peekLast: 67
----------------------


pop
printing deque...
dequesize: 7
4456 169 463 169 1 169 67
printing removed list...
45 100
getFirst:4456
peekFirst: 4456
getLast:67
peekLast: 67
----------------------


remove
printing deque...
dequesize: 6
169 463 169 1 169 67
printing removed list...
4456 45 100
getFirst:169
peekFirst: 169
getLast:67
peekLast: 67
----------------------
```

```
removeLastOccurence 169
printing deque...
dequesize: 5
169 463 169 1 67
printing removed list...
169 4456 45 100
getFirst:169
peekFirst: 169
getLast:67
peekLast: 67
-----------------------

remove Last:
printing deque...
dequesize: 4
169 463 169 1
printing removed list...
67 169 4456 45 100
getFirst:169
peekFirst: 169
getLast:1
peekLast: 1
-----------------------

poll Last:
printing deque...
dequesize: 3
169 463 169
printing removed list...
1 67 169 4456 45 100
getFirst:169
peekFirst: 169
getLast:169
peekLast: 169
-----------------------
```

Q3)

Problem Solutions Approach

Q1)

***Problem**: Reversing a string. For example, if the input is "this function writes the sentence in reverse", then the output should be "reverse in sentence the writes function this".*

***Solution**: In my main reverseString method, firstly I split the string words by words with using String's split method. After I send this Words array to my helper function. In my helper function, I add the last Word and one space to the begining of the output. I control this with index. Every method call,index is decreasing 1. When index is 0, helper method's mission is completed. Then my main reverseString method returns the output.*

Q2)

***Problem**: Determining whether a word is elfish or not. A word is considered elfish if it contains the letters: e, l, and f in it, in any order. For example, whiteleaf, tasteful, unfriendly, and waffles are some elfish words.*

***Solution**: My base case is given input is null or not. If I reach end of the input, I controlled eNum,lNum and fNum which are how many time I found the letters. If all of the parameters are bigger than 1, return method returns true,otherwise false.*

If I did not reach the end of the input, I control that current index is which letter. If I found letter which I looking for,I increase that letter's number and returns the input with substring(1).

Q3)

***Problem**: Sorting an array of elements using selection sort algorithm.*

Solution:

For solve and showing this problem,I write three methods. First method for finding the minimum value's index which is started given index. That method returns the minimum value's index. Method's name: findMinIndex();

My second method is print array method which is printing the array recursively. Method's name: printArray();

My main method for selection sort is selectionSort(). That method is firstly controlling the index is equals to size-1 or not. If it is equals, then returns the array. Which means we reach the end of the array. Otherwise, firstly we call the findMinIndex method and find the minimum index. Then compare the value of array's current position and minimum index position. If minimum index position is smaller than our current position, then swap the values. Returns the main method again with increase 1 the index number.

If our base case is supplied, method call is completed and we reach the solution.

Q4)

Problem : *Evaluating a Prefix expression*

Solution:

For that problem, I write three method. First method is isDigit(). Which is controling the character is digit or not. İf character is digit,then returns true, otherwise false.

My second method is isOperator() which is controlling the character is operator or not. If character is operator returns true, otherwise false.

My third and main method is evaluatePrefix(). That method start to read from end of the expression character by character. If character is equals to number, assign to temp. After the assigning the next character is also number, then calculate the digit value all of them. For doing this, I keep the digit value as a parameter. But it should be start 0. And after every space character, I set to zero that value and number which is holding in temp value pushed to stack. That method returning the finding a operator. When operator found, values poped from the stack and done evaluation and push again the result to the stack. And every method call, index is decrease 1. All the recursion calls end if index is smaller than 0. And that case is my base case.

Q5)

Problem: *Evaluating a Postfix expression*

Solution: *For this problem, I use 3 method. Two of them is written before for the prefix expression which are isDigit() and isOperator().*

*In main method, I start to read from beginning of the expression character by character. If character which I read is number, I assign to temp that character's integer equivalent. In next recursive call, character is number again, I add to temp with that digit value.(for example 10*2). Until I found a space character, I do that steps. İf operand size is not 2, when I found the space, I push the temp to the stack. İf operand size is equals to 2, then I continue. If I found operator, I pop the two values of the stack and do evaluation and call the recursive call with decrease 1 the index number. If index number is equal to expression's size, recursive calls are completed and method returns the evaluation.*

Q6)

Problem : *Printing the elements of an array in different way.*

Solution:

My base condition is counter is equal to total size or not. Which means I travel all the numbers or not. İf I travel all numbers, that recursive call is completed. My other conditons:

My first condition is current position is on top row and it can continiue to right side. If we are in that condition, then

returns recursive call with increase one the number of index of column.

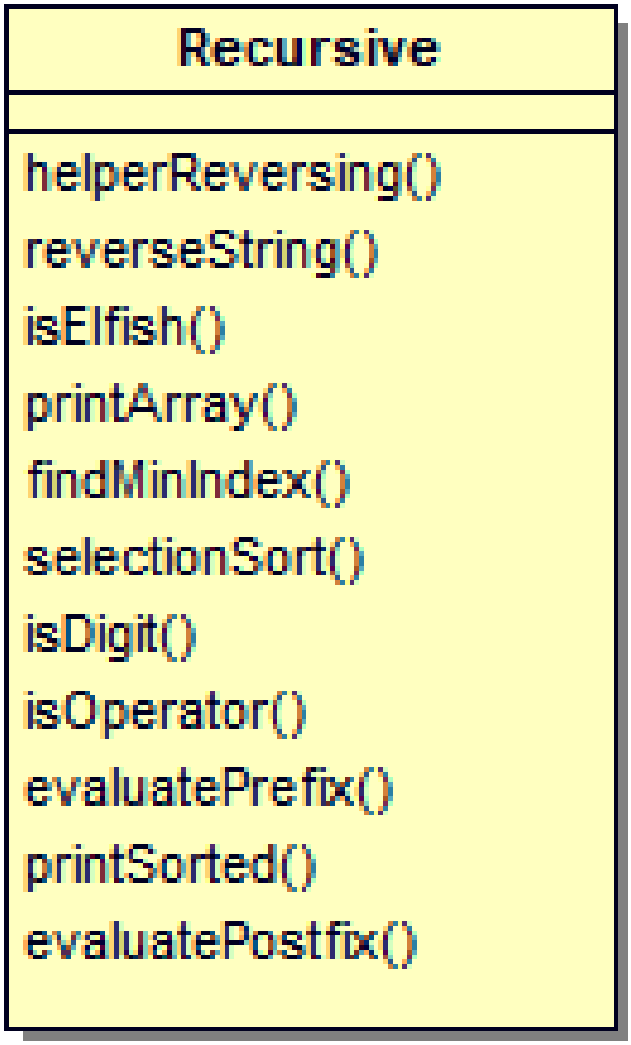
My second condition is current position is on bottom row and it can continue to left side. If we are in that condition, then returns recursive call with decrease one the number of index of column.

My third condition is current position is mostLeft column but it wont turn right yet. If we are in that condition, then returns recursive call with decrease one the number of index of row.

My fourth condition is current position is mostleft column and it will turn to the right side at that time. If we are in that condition, , then returns recursive call with increase one the start row,decrease one end row, increase one the start column, decrease one the end column and increase one the index of column.

My fifth condition is current position is mostright column. If we are in that condition, , then returns recursive call with increase one the index of row.

Class Diagram



Test cases and results

Question 1: *Reversing a string. For example, if the input is "this function writes the sentence in reverse", then the output should be "reverse in sentence the writes function this".*

```
Question 1: Reversing a string.

String 1:  this function writes the sentence in reverse
Reverse : reverse in sentence the writes function this

String 1:  Elif Levent Neslihan Abdullah
Reverse : Abdullah Neslihan Levent Elif

String 1:
java.lang.Exception: String is empty
Reverse :

------------------------------------------------------------
```

Question 2: *Determining whether a word is elfish or not. A word is considered elfish if it contains the letters: e, l, and f in it, in any order. For example, whiteleaf, tasteful, unfriendly, and waffles are some elfish words*

```
------------------------------------------------------------
Question 2: Determining whether a word is elfish or not
 A word is considered elfish if it contains the letters: e, l, and f in it, in any order

String : unfriendly -> true
String : whiteleaf -> true
String : tasteful -> true
String : waffles -> true
String : elif -> true
String : levent -> false
String : asdf -> false
String : qwert -> false
```

Question 3: *Sorting an array of elements using selection sort algorithm.*

```
------------------------------------------------------------
Question 3: Sorting an array of elements using selection sort algorithm.

array1 which will be sorted:
3 5 8 4 1 9 -2
Selection sort algorithm started...
-2 5 8 4 1 9 3
-2 1 8 4 5 9 3
-2 1 3 4 5 9 8
-2 1 3 4 5 9 8
-2 1 3 4 5 9 8
-2 1 3 4 5 8 9
selection sorted array1:
-2 1 3 4 5 8 9

array2 which will be sorted:
1 8 221 95 -9 12 65
Selection sort algorithm started...
-9 8 221 95 1 12 65
-9 1 221 95 8 12 65
-9 1 8 95 221 12 65
-9 1 8 12 221 95 65
-9 1 8 12 65 95 221
-9 1 8 12 65 95 221
selection sorted array 2:
-9 1 8 12 65 95 221

------------------------------------------------------------
```

Question 4: *Evaluating a Prefix expression*

```
Question 4: Evaluating a Prefix expression

Prefix Expression: - / * 20 * 50 + 3 6 300 2
2
temp:2
[2]
0
0
3
temp:300
[2, 300]
6
temp:6
[2, 300, 6]
3
temp:3
[2, 300, 6, 3]
3 + 6 = 9
[2, 300, 9]
0
5
temp:50
[2, 300, 9, 50]
[2, 300, 450]
50 * 9 = 450
0
2
temp:20
[2, 300, 450, 20]
[2, 300, 9000]
20 * 450 = 9000
[2, 30]
9000 / 300 = 30
[28]
30 - 2 = 28
Evaluation: 28
```

```
Prefix Expression: + * / 135 * 5 + 3 6 120 2
2
temp:2
[2]
0
2
1
temp:120
[2, 120]
6
temp:6
[2, 120, 6]
3
temp:3
[2, 120, 6, 3]
3 + 6 = 9
[2, 120, 9]
5
temp:5
[2, 120, 9, 5]
[2, 120, 45]
5 * 9 = 45
5
3
1
temp:135
[2, 120, 45, 135]
[2, 120, 3]
135 / 45 = 3
[2, 360]
3 * 120 = 360
360 + 2 = 362
[362]
Evaluation: 362
```

Question 5 : *Evaluating a Postfix expression*

```
------------------------------------------------------------
Question 5: Evaluating a Postfix expression.
postfix expression: 100 200 + 2 / 5 * 7 +
1
0
0
2
0
0
100 + 200 = 300
2
300 / 2 = 150
5
150 * 5 = 750
7
750 + 7 = 757
757

postfix expression: 40 20 - 2 / 4 * 5 -
4
0
2
0
40 - 20 = 20
2
20 / 2 = 10
4
10 * 4 = 40
5
40 - 5 = 35
35
```

***Question 6**: Printing the elements of an array in different way.*

```
------------------------------------------------------------
Question 6: Printing the elements of an array different type.

Our array is:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20

sorted way:
1 2 3 4 8 12 16 20 19 18 17 13 9 5 6 7 11 15 14 10

Our array is:
1 2 3 4 5 6
7 8 9 10 11 12
13 14 15 16 17 18
19 20 21 22 23 24
25 26 27 28 29 30

sorted way:
1 2 3 4 5 6 12 18 24 30 29 28 27 26 25 19 13 7 8 9 10 11 17 23 22 21 20 14 15 16

------------------------------------------------------------
```