Name : Fardouse Jonat Jahan Rumpa

ID : 2219170091

Dept : B. Sc in CSE

Course title : Theory of computing

Course code : CSI 317

Semester : Summer 2023

Ans to the Qus No: 01 (a)

Ans: Insertion in a queue follows a straight forward algorithm that adds an element to the back of the queue. Here's the step-by-step algorith for inserting an element into a queue.

Algorithm for inserting in Queue.

(1) Check if the Queue is full. (if applicable):

*If the queue has maximum capacity. Check whether the queue is already full. If it is full, the inserting cannot be performed. Handle this case based on the requirements of your application.

(2) Create a New Node or Element:

Create a new node or element that contains the data you want to insert into the queue.

(3) Set the New Node's Data:

Assign the data or value you want to insert into the new node.

(4) Update Pointers:

if your queue implementation maintains a rear pointer, update the rear pointer to point to the new node.

⑤ Update rear Pointer:

If your queue implementation maintains a rear pointer, update the rear pointer to point to the new node.

⑥ Update queue Size:

If you are tracking the size of the queue, increment the size counter by 1.

The insertion algorithm ensures that the new element becomes the last element in the queue; following the first-in-first out (FIFO) principle. Elements are always inserte at the rear and removed from the front in queue data structure.

Here's simplified version of the insertio algorithm in pseudocode:

Algorithm Insertion in Queue:

Input: Data to be inserted.

1. Create a new node with the given data.
2. if the queue is empty, set both front and rear pointers to the new node.

3. Else, set the next pointer of the current rear node to point to the new node.
4. Update the rear pointer to the new node.
5. If the queue size is tracked, increment the size counter by 1.
6. End Insertion Algorithm.

This algorithm ensures that the new element is properly inserted into the queue, maintaining the queue's structure and properties.

Ans to the QUS NO: 01 (b)

Ans: Algorithm to traverse a Singly Linked List:

→ Step 01: start

→ Step 02: [Declare a pointer named temp of node type.]

→ Step 03: set temp = head

→ Step 04: if head = = NULL

→ Step 05: Display "List is empty"

→ Step 06 : Go to Step 11

→ Step 07 : [End of if]

→ Step 08 : Else.

→ Step 09 : Display temp -> data

→ Step 10 : Repeat Step 10 and 11 until temp != NULL.

→ Step 11 : Set temp = temp -> next.

→ Step 12 : [End of step 09 loop.]

→ Step 13 : [End of Else block.]

→ Step 14 : Stop.

Ans to the Qus No:02 (a)

Ans: Converting an infix expression to a postfix expression involves using the Shunting Yard algorithm. Here are the steps for converting the given infix expression,

A+(B*C-(D/E^F)*G) into postfix notation

Lets take an example to better understand the algorithm:

Input String	Output Stack	Operator Stack.
A+(B*C-(D/E^F)*G)*H		
A+(B*C-(D/E^F)*G)*H	A	
A+(B*C-(D/E^F)*G)*H	A	+
A+(B*C-(D/E^F)*G)*H	A	+(
A+(B*C-(D/E^F)*G)*H	AB	+(
A+(B*C-(D/E^F)*G)*H	ABC	+(*
A+(B*C-(D/E^F)*G)*H	ABC*	+(*
A+(B*C-(D/E^F)*G)*H	ABC*	+(-
A+(B*C-(D/E^F)*G)*H	ABC*D	+(-(
A+(B*C-(D/E^F)*G)*H	ABC*D	+(-(
A+(B*C-(D/E^F)*G)*H	ABC*DE	+(-(/
A+(B*C-(D/E^F)*G)*H	ABC*DE	+(-(/
A+(B*C-(D/E^F)*G)*H	ABC*DEF	+(-(/^
A+(B*C-(D/E^F)*G)*H	ABC*DEF^/	+(-(/^
A+(B*C-(D/E^F)*G)*H	ABC*DEF^/	+(-
A+(B*C-(D/E^F)*G)*H	ABC*DEF^/G	+(-*
A+(B*C-(D/E^F)*G)*H	ABC*DEF^/G*-	+(-*
A+(B*C-(D/E^F)*G)*H	ABC*DEF^/G*-H	+
A+(B*C-(D/E^F)*G)*H	ABC*DEF^/G*-H	+*
A+(B*C-(D/E^F)*G)*H	ABC*DEF^/G*-H*	+*
A+(B*C-(D/E^F)*G)*H	ABC*DEF^/G*-H*+	

Advantage of Postfix Expression over Infix Expression:

→ After scaning "A+". A is added to the output.
→ Upon encountering "(". + is pushed onto the operator stack.
→ Upon encountering "B*", B is added to the output.
→ Upon encountering "C". C is already to the output.
→ Upon encountering ")", operations * and B are popped from the stack and added to the output.
→ Upon encountering "-", operator - is pushed onto the
→ Upon encountering "(", + is pushed onto the stack.
→ Upon encountering "D", D is added to the output.
→ Upon encountering "/". operator / and D are added to the output.

→ Upon encountering "E^F", E, F, ^, and / are added to the output.

Ans to the Qus No: 03(a)

Ans: Here's the step by-step algorithm for converting an infix expression into a postfix expression using the shunting yard algorithm.

Algorithm for Infix to Postfix conversion (Shunting Yard Algorithm)

① Initialize two stacks.

→ One for operators (operator stack)
→ One for the output expression (output stack or queue)

② Start Scaning the infix Expression from Left to Right:

→ For each token in the infix expression:
→ If the token is an operand (like A→B ,1,2 ,etc) add it to the output stack or queue.

→ if the token is a right parenthesis ")". pop operators from the operator stack and add them to the output stack or queue until a left parenthesis is encountered. Pop and discard the left parenthesis.

→ if the token is an operator:
→ While the operator stack is not empty and the top of the operator stack has higher or equal precedence. compared to the current

operator, and the current operator is left associative pop the operator from the operator stack or queue.

→ Push the current operator onto the operator stack.

③ After scanning the Entire Expression:

→ pop any remaining operators from the operator stack and add them to the output stack or queue.

④ End of Conversion:

The output stack or queue now contains the postfix expression.

Example:

Consider the infix expression: A+B*(C-D)/E

→ Using the shunting Yard algorithm, the postfix expression will be: ABCD-*E/+

This algorithm ensures the correct conversion of infix expression into postfix expressions preserving the operator precedence and associativity rules.

Ans to the Qus No: 04 (a)

Ans: Searching in Singly Linked List:

Searching is performed in order to find the location of a particular element in the list. Searching any element in the list needs traversing through the list and make the comparison of every element of the list with the specified element. If the element is matched with any of the list element is returned from the function.

Algorithm:

→ Step 1 : SET PTR = HEAD

→ Step 2 : Set I = 0

→ Step 3 : IF PTR = NULL

→ ~~Step~~ WRITE "EMPTY LIST"

GOTO STEP 8

END OF IF

→ Step 4 : REPEAT STEP 7 TO 7 UNTIL PTR != NULL

→ Step 5 : if Ptr → data = item

write i + 1

End of IF

→ Step 6: I = I + 1

→ Step 7 : PTR = PTR → NEXT

[END of LOOP]

→ Step 8: EXIT.

This algorithm ensures the correct conversion of infix expressions into postfix expressions, preserving the operators precedence and associativity rules.