(Q3) a) Infix to Postfix.

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

Symbol	Stack.	Postfix.
(	(	
A	(	A
+	(+	A
B	(+	A B
*	(+ *	A B
(	(+ * (	A B
C	(+ * (	A B C
-	(+ * (-	A B C
D	(+ * (-	A B C D
)	(+ *	A B C D -
^	(+ * ^	A B C D -
E	(+ * ^	A B C D - E
)		A B C D - E ^ * +
/	/	A B C D - E ^ * +

Question Nos. | Marks Awarded

Symbol	Stack	Postfix
(	(	
(	((	
F	((	F
-	((-	F
G	((-	F G
+	((+	F G -
H	((+	F G - H
*	((+ *	F G - H
I	((+ *	F G - H I
)	(	F G - H I * +

So the postfix expression is

F G - H I * +

Question Nos.

Q2) b) Circular Linked List.

(1) Insertion at the start.

Step 1 : Set ptr = start

Step 2 : If ptr = NULL
ptr = New node
~~New node = start~~
New node → next = start.
exit.

Step 3 : If start != NULL.

Explanation.

~~the~~ we point the head to the new node. & then point the new nodes next ptr to the first node.

~~this boy~~

Question Nos. | Marks Awarded

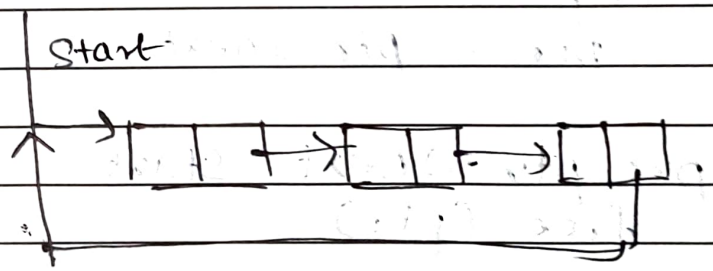
(Q2) b) Circular Linked List.

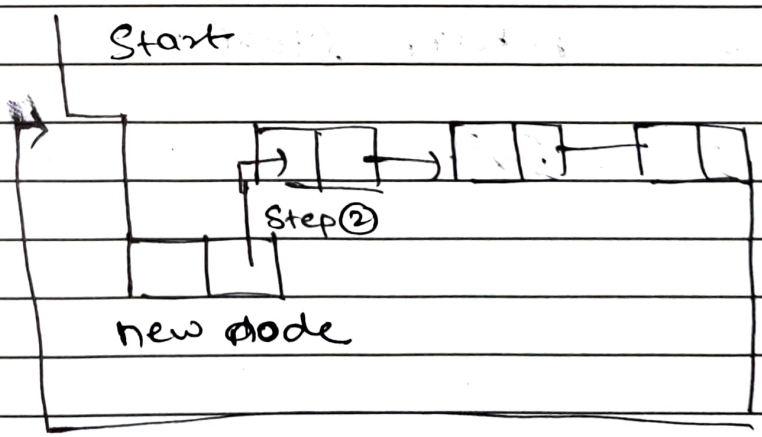
(1) Insertion at the start.

Step 1 : Set ptr = ~~new~~ start (head)
Input n = data.

Step 2 : Set new node → data = n.
new node → next = start
ptr = new node.

Step 3 : ~~exit~~. return start
Step 4 : exit
Example.





Question Nos.

2.) Deletion of last node.

Step ① : Set ptr = ~~head~~ Start
pre ptr = ptr
Input n = data..

Step ② : if ptr = NULL
print ("List is Empty").
Exit. Step 6.

Step ③ : else ~~if ptr~~
~~do while (ptr != start)~~
do while (ptr != start)
preptr = ptr
ptr = ptr → next

Step ④ : pre ptr → next = start
free (ptr)

Step ⑤ : return start.

Step ⑥ : Exit.

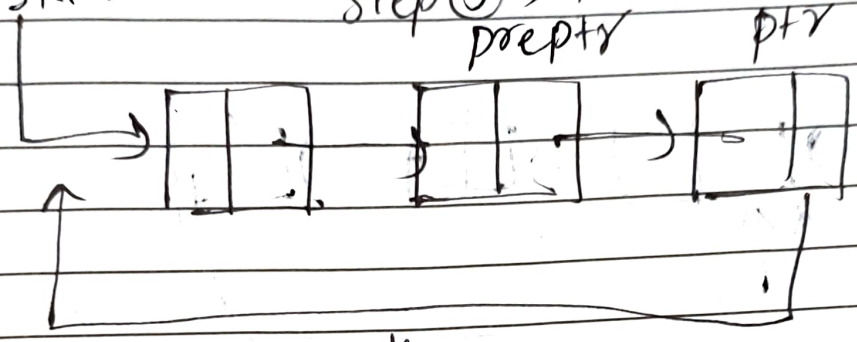
Question Nos.

Marks Awarded

Example.

Start.

Step (3) ⇒ traversed till the end.



~~Step (4)~~

we used do while (ptr != ~~Null~~ start) because there are two instances when ptr = start once in the start & once in the end. & what do while does is it atleast runs once so we pass the first instance when we got ptr = ~~Null~~ start.

next instance when ptr = start is at the end of the list. this when the do while loop breaks.

we converge the prepointer to start & free the ptr which is at the end of the node.

removing the last node from the list.

Q1) Asymptotic notations.

→ big O of n.

O(n) : worst case.

O(n) represents the worst case scenario ie the algorithm can't take more time than this case. maximum time the algorithm will take to run. Highest time complexity

→ big Omega of n.

Ω(n) : Best case.

Ω(n) represents the best case. ie the algorithm / code won't take lesser time in any other case. minimum time / fastest algorithm can take. Lowest time complexity eg while sorting if you already are given a sorted list.

→ bige theta of n.

Θ(n) : Average case.

Θ(n) represents the average case ie the algorithm will take avg time.