CHAPTER-10
LINKED LISTS, STACKS AND QUEUES
VERY SHORT/SHORT ANSWER QUESTIONS

1. **Translate, following infix expression into its equivalent postfix expression:**
((A-B)*(D/E))/(F*G*H)

Ans. Equivalent postfix expression:
=((A-B)*(D/E))/(F*G*H)
=((AB-)*(DE/))/(FG*H*)
=AB – DE /* FG* H*/

2. **Translate, following infix expression into its equivalent postfix expression:**
(A+B↑D)/(E-F)+G

Ans. (A+[BD↑])/[EF-]+G
=[ABD↑+]/[EF-]+G
=[ABD↑+EF-/]+G
=ABD↑+EF-/G+

3. **Translate, following infix expression into its equivalent postfix expression:**
A*(B+D)/E-F-(G+H/K)

Ans. Equivalent postfix expression:
= A*(B+D)/E-F-(G+H/K)
= (A*(B+D)/E) – (F – (G + (H/K)))
= (A*(BD+)/E) – (F – (G + (HK/)))
= ((ABD+*) / E) – (F – (GHK/+))
= ABD+* E/F – GHK/+ -

4. **Write the equivalent infix expression for**
10,3,*,7,1,-,*,23,+

Ans. 10 * 3 * (7 – 1) + 23

5. **Write the equivalent infix expression for a, b, AND, a, c, AND, OR.**

Ans. a, b, AND, a, c, AND, OR
(a AND b), (a AND c), OR
(a AND b) OR (a AND c)

6. **Consider the arithmetic expression P, written in postfix notation:**
12,7,3,-,/,2,1,5,+,*,+
(a) Translate P, into its equivalent infix expression.
(b) Evaluate the infix expression.

Ans. **(a) Consider the following arithmetic expression P, written in postfix notation:**
P: 12, 7, 3, - , /, 2, 1, 5 , + , * , +
Translate P into infix expression.
P: 12,[7-3],/,2,1,5,+,*,+
=[12/(7-3)],2,1,5,+,*,+
=[12/(7-3)],2,[1+5],*,+
=[12/(7-3)],[2*(1+5)],+
=12/(7-3)+2*(1+5)

(b) Evaluate the infix expression.
P: 12 , 7 , 3 , - , / , 2 , 1 , 5 , + , * , + ,)

Symbol	Stack
12	12
7	12,7
3	12,7,3
-	12,4
/	3
2	3,2

1	3,2,1
5	3,2,1,5
+	3,2,6
*	3,12
+	15
)	15

7. **Convert the following infix notation of expression to an equivalent postfix notation of expression (Show status of stack after execution of each operation): (A+B)*C-D/E**

Ans. Conversion of (A+B)*C-D/E into postfix expression

Step	Input element/ symbol taken	Action	Stack status	Output to be printed
			#(empty)	
1.	'('	PUSH	#'('	
2.	'A'	Print	#'('	A
3.	'+'	PUSH	#'(+	A
4.	'B'	Print	#'(+	A
5.	')'	POP & Print POP	#'(# #	AB+
6.	'*'	PUSH	#*	AB+
7.	'C'	Print	#*	AB+C
8.	'-'	PUSH	#*-	AB+C
9.	'D'	Print	#*-	AB+C*D
10.	'/'	PUSH	#*-/	AB+C*D
11.	'E'	Print	#*-/	AB+C*DE/
12.	;	POP & Print STOP	#(empty)	AB+C*DE/-

The resultant expression is AB+C*DE/-

8. **Consider each of the following postfix expressions:**
P1 : 5, 3, +, 2, *, 6, 9, 7, -, /, -
P2 : 3, 5, +, 6, 4, -, *, 4, 1, -, 2, ↑, +
P3 : 3, 1, +, 2, ↑, 7, 4, -, 2, *, +, 5, -
Translate each expression into infix notation and then evaluate.

Ans. P1: (((5+3)*2)-(6/(9-7)))
(8*2)-(6/2)
16-3
13
P2: (((3+5)*(6-4))+((4-1))^2))
((8*2)+3^2))
(16+3^2)
16+9
25
P3: Students why don't you try at least one.

9. **Give postfix form of the following expression A*(B+(C+D)*(E+F)/G)*H**

Ans. A*(B+(CD+EF+*)/G)*H
A*(B+CD+EF+*G/))*H
(A*(BCD+EF+*G/+))H
(ABCD+EF+*G/+*)*H
ABCD+EF+*G/+*H*

10. **Give postfix form for A+[(B+C)+(D+E)*F]/G**

Ans.

A+[{B+C)+((D+E)*F)}/G]
=A+[{(BC+)+(DE+)*F}/G]
=A+[{(BC+)+(DE+F*)}/G]
=A+[{(BC+DE+F*+}/G]
=A+[BC+DE+F*+G/]
=ABC+DE+F*+G/+

11. **Give postfix form expression for the following:**

(i) **NOT A OR NOT B AND NOT C**

(ii) **NOT (A OR B) AND C**

Ans.

(i) =((A NOT) OR ((B NOT) AND (C NOT)))
=((A NOT) OR ((B NOT C NOT AND))
=A NOT B NOT C NOT AND OR

(ii) Try this and see your progress.

12. **Evaluate the following postfix notation of expression: True, False, NOT, AND, True, True, AND, OR**

Ans.

Step	Input Symbol	Action Taken	Stack status	Output
1.	True	Push	True (↑ - top) ↑	
2.	False	Push	True, False ↑	
3.	NOT	Pop (1 element) Push result (True)	True ↑ True, True ↑	NOT False = True
4.	AND	Pop (2 element) Push result (True)	↑ (empty stack) True ↑	True AND True = True
5.	True	Push	True, True ↑	
6.	True	Push	True, True, True ↑	
7.	AND	Pop (2 element) Push result (True)	True ↑ True, True ↑	True AND True = True
8.	OR	Pop (2 element) Push result (True)	↑ (empty stack) True ↑	True OR True = True
			Ans = True	

13. **Write an algorithm to convert infix expression to postfix expression.**

Ans.

1. Push "(" onto STACK, and add ")" to the end of X.
2. Scan X from left to right and REPEAT Steps 3 to 6 for each element of X UNTIL the STACK is empty.
3. If an operand is encountered, add it to Y.
4. If a left parenthesis is encountered, push it onto STACK.
5. If an operator is encountered, then:
   (a) Repeatedly pop from STACK and add to Y each operator (on the top of STACK) which has the same precedence as or higher precedence than operator.
   (b) Add operator to STACK.
   /* End of If structure */
6. If a right parenthesis is encountered, then:
   (a) Repeatedly pop from STACK and add to Y each operator (on the top of STACK) until a left Parenthesis is encountered.

(b) Remove the left parenthesis. (Do not add the left parenthesis to Y).
/* End of If structure */
7. END.

14. **Write an algorithm to insert element into (i) Stack as an array (ii) Linked stack.**

Ans. **Stack as an array**

/* Assuming that array stack can hold maximum N elements
1. Top=-1
2. Read Item //Item, which is to be pushed
3. If (top==N-1) then
4. {Print "overflow!!"
}
5. Else
6. { top = top + 1
7. Stack [top]=Item }
8. END

Linked stack.

/*Get new node for the ITEM to be pushed*/
1. NEWPTR = new Node
2. NEWPTR ->INFO = ITEM;
NEWPTR->LINK=NULL
/*Add new node at the top*/
3. If TOP = NULL then /*stack is empty*/
4. TOP=NEWPTR
5. Else
6. { NEWPTR ->LINK=TOP
7. TOP=NEWPTR
} //end of step 3
8. END.

15. **Evaluate the following postfix expression using a stack and show the contents of the stack after execution of each operation 5, 11, -, 6, 8, +, 12, *, /**

Ans.

SNo.	Symbol	Action taken	Stack	Intermediate output
1	5	operand, push	5	
2	11	operand, push	5,11	
3	-	operator, pop twice		5-11=-6
		Evaluate and push back	6	
4	6	operand, push	6,-6	
5	8	operand, push	6,-6,8	
6	+	operator, pop twice	6	-6+8=-14
		Evaluate and push back	6,14	
7	12	operand, push	6,14,12	
8	*	operator, pop twice	6	12*14=168
		Evaluate and push back	6,168	
9	/	operator, pop twice		168/6
10		pop all	-1/28 Ans.	

16. **Suppose a queue is maintained by a circular array QUEUE with N=12 memory cells. Find the number of elements in QUEUE if (a) FRONT = 4, REAR = 8; (b) FRONT = 10 REAR = 3; and (c) FRONT = 5, REAR = 6 and then two elements are deleted.**

Ans. Try by yourself

17. **Let P be the postfix arithmetic expression: 7, 2, -, 1, 14, -, 1, 2, * Evaluate P using stack and showing the status of the stack at every step.**

Ans.

SNo.	Symbol	Action taken	Stack	intermediate output

1	7	operand, push	7	
2	2	operand, push	7, 2	
3	-	operator, pop twice		7-2=5
		Evaluate and push back	5	
4	1	operand, push	5, 1	
5	14	operand, push	5, 1, 14	
6	-	operator, pop twice	5	1-14=-13
		Evaluate and push back	5, -13	
7	1	operand, push	5, -13, 1	
8	2	operand, push	5, -13, 1, 2	
9	*	operator, pop twice	5, -13	1*2=2
		Evaluate and push back	5, -13, 2	
As 2 is the Top most value so answer should be 2.				

18. **Consider the infix expression**
Q : A+B * C ↑ (D/E)/F.
Translate Q into P, where P is the postfix equivalent expression of Q. what will be the result of Q if this expression is evaluated for A, B, C, D, E, F as 2, 3, 2, 7, 2, 2 respectively.

Ans.
P = ABCDE/^*F/+
2,3,2,7,2
2,3,2->7/2->3
2,3,2,3
2,3->2^3->8
2,3,8
2->3*8->24
2,24,2
2->24/2->12
2,12
2+12
Result of evaluation = 14

19. **Differentiate between**
(i) Linear Queue and Circular Queue
(ii) A Queue and a Dequeue.

Ans.
(i) <u>Linear Queue</u>
1. A linear queue is like a straight line in which all elements or instructions stand one behind the other.
2. Tasks lined up in this queue format are executed in the order of their placement, on a FIFO (First In First Out) basis.
3. In a linear queue, a new task is inserted at the end of the list, while a deletion is made at the front of the list.
4. Allocate new, larger block of memory reflective of the entire (existing) queue and the queue-size additions.

<u>Circular Queue</u>
1. A circular queue has a circular structure. The last element of this queue is connected with the first element, thus completing the circle.
2. Tasks in this format are not essentially executed in the order in which they are sent.
3. In a circular queue, insertions and deletions can happen at any position in the queue.
4. Allocate ONLY the new object or structure-type required.

(ii) <u>A Queue and dequeue</u>
A **queue** is a linear structure implemented in FIFO (First In First Out) manner where insertions can occur only at the "rear" end and deletion can occur only at the "front" end whereas **Dequeues** are the refined queues in which elements can be added or removed at either end but not in the middle.

20. **Write an algorithm to insert an ITEM in a QUEUE implemented as an array.**

Ans.
1. if rear = NULL Then
{

	2. rear = front=0
	3. QUEUE[0] = ITEM
	}
	4. Else If rear = N-1 Then
	5. Print "QUEUE FULL, OVERFLOW!!"
	6. Else
	{
	7. QUEUE[rear+1]=ITEM
	8. rear=rear+1
	9. END.

21.	**Write an algorithm to delete an ITEM from a QUEUE implemented as an array.**

Ans.	1. If front = Null Then
	2. Print "Queue Empty"
	3. Else
	{
	4. ITEM=QUEUE[front]
	5. If front=rear Then
	{
	6. front = rear = NULL
	}
	7. Else
	8. front=front+1
	} /* end of step 3*/
	9. END.

22.	**Write an algorithm to insert an ITEM in a QUEUE implemented as a linked list.**

Ans.	1. NEWPTR = new Node
	2. NEWPTR ->INFO = ITEM;
	NEWPTR->LINK=NULL
	/*insert in the queuep*/
	3. If rear = NULL then
	{
	4. Front = NEWPTR
	5. Rear = NEWPTR
	}
	6. Else
	{
	7. Real ->LINK= NEWPTR
	8. rear=NEWPTR
	}
	9. END.

23	**Write an algorithm to delete an ITEM from a QUEUE implemented as a linked list.**

Ans.	1. If front = Null Then
	2. Print "Queue Empty"
	3. Else
	{
	4. ITEM=front-> INFO
	5. If front=rear Then
	{
	6. front = rear = NULL
	}
	7. Else
	8. front=front->LINK

}
9. END.

24. **Write equivalent postfix prefix and expressions for the infix expression given below:**
(a) A+B-D/X
(b) (X+Y)/(Z*Y)-R

Ans.
```
Postfix: (i) AB+DX /-          (ii) XY+ZY*/R-
Prefix   (i) -+AB/DX           (ii) -/+XY*ZYR
```

25. **Evaluate the following postfix notation of expression: 10 20 + 25 15 - * 30 /**

Ans.

SNo.	Symbol	Action taken	Stack	intermediate output
1	10	operand, push	10	
2	20	operand, push	10, 20	
3	+	operator, pop twice		10+20=30
		Evaluate and push back	30	
4	25	operand, push	30, 25	
5	15	operand, push	30, 25, 15	
6	-	operator, pop twice	30	25-15=10
		Evaluate and push back	30, 10	
7	*	operator, pop twice		30*10=300
		Evaluate and push back	300	
8	30	operand, push	300, 30	
9	/	operator, pop twice		300/30=10
10		Evaluate and push back	10	
11		pop all	**10 Ans.**	

26. **Evaluate the following postfix notation of expression: 20 10 + 5 2 * - 10 /**

Ans.

SNo.	Symbol	Operation	Stack	Intermediate output
1	20	Push	20[↑signifies top] ↑	
2	10	Push	20, 10 ↑	
3	+	Pop twice calculate	#	20+10=30
		Push the result	30 ↑	
4	5	Push	30, 5 ↑	
5	2	Push	30, 5, 2 ↑	
	*	Pop twice calculate	30 ↑	5 * 2 = 10
		Push the result	30, 10 ↑	
6	-	Pop twice calculate	#	30 – 10 = 20
		Push the result	20 ↑	
7	10	Push	20, 10 ↑	
	/	Pop twice calculate	#	20 / 10 = 2
		Push the result	2	
8	End of	Pop	2	

		Expression			
				Ans. = 2	

27. **Evaluate the following postfix expression using a stack and show the contents of stack after execution of each operation: 120, 45, 20, +, 25, 15, -, +, ***

Ans.

SNo.	Symbol	Action taken	Stack	intermediate output
1	120	operand, push	120	
2	45	operand, push	120, 45	
3	20	operand, push	120, 45, 20	
4	+	operator, pop twice	120	45+20=65
		Evaluate and push back	120, 65	
5	25	operand, push	120, 65, 25	
6	15	operand, push	120, 65, 25, 15	
7	-	operator, pop twice	120, 65	25-15=10
		Evaluate and push back	120, 65, 10	30*10=300
8	+	operator, pop twice	120	65+10=75
		Evaluate and push back	120, 75	
9	*	operator, pop twice		120*75=9000
		Evaluate and push back	9000	
10		pop all	**9000 Ans.**	

28. **Evaluate the following postfix expression using a stack and show the contents of stack after execution of each operation: 20, 45, +, 20, 10, -, 15, +, ***

Ans.

SNo.	Symbol	Action taken	Stack	intermediate output
1	20	operand, push	20	
2	45	operand, push	20, 45	
3	+	operator, pop twice		20+45=65
		Evaluate and push back	65	
4	20	operand, push	65, 20	
5	10	operand, push	65, 20, 10	
6	-	operator, pop twice	65	20-10=10
		Evaluate and push back	65, 10	
7	15	operand, push	65, 10, 15	
8	+	operator, pop twice	65	10+15=25
		Evaluate and push back	65, 25	
9	*	operator, pop twice		65*25=1625
		Evaluate and push back	1625	
10		pop all	**1625 Ans.**	

29. **Use a stack to evaluate the following postfix expression and show the content of the stack after execution of each operation. Don't write any code. Assume as if you are using push and pop member functions of the stack AB-CD+E*+**
(where A=5, B=3, C=5, D=4, and E=2

Ans. Expression is 5, 3, -, 5, 4, +, 2, *, +

SNo.	Symbol	Action taken	Stack	intermediate output
1	5	operand, push	5	
2	3	operand, push	5, 3	
3	-	operator, pop twice		5-3=2
		Evaluate and push back	2	
4	5	operand, push	2, 5	
5	4	operand, push	2, 5, 4	
6	+	operator, pop twice	2	5+4=9
		Evaluate and push back	2, 9	

7	2	operand, push	2, 9, 2	
8	*	operator, pop twice	2	9*2=18
		Evaluate and push back	2, 18	
9	+	operator, pop twice		2+18=20
		Evaluate and push back	20	
10		pop all	**20 Ans.**	

30. **Change the following infix expression into postfix expression. (A+B) *C+D/E-F**

Ans. Equivalent postfix expression:
= (A+B) *C+D/E-F
= (((A+B)*C) + (D/E)) – F
= (((AB+)*C) + (DE/)) – F
= AB+ C* DE/ + F-

31. **Evaluate the following postfix notation of expression: (Show status of stack after each operation)**
True, False, NOT, OR, False, True, OR, AND

Ans.

Element Scanned	Stack
True	True
False	True,False
NOT	True,True
OR	True
False	True,False
True	True,False,True
OR	True,True
AND	True

32. **Convert the following infix expression to its equivalent postfix expression showing stack contents for the conversion:**
X - Y / (Z + U) * V

Ans. X - Y / (Z + U) * V
= (X - ((Y / (Z + U)) * V))

Element Scanned	Stack	Postfix
(		
X		X
-	-	
(		
(		
Y		XY
/	-/	
(		
Z		XYZ
+	-/+	
U		XYZU
)	-/	XYZU+
)	-	XYZU+/
*	-*	
V		XYZU+/V
)	-	XYZU+/V*
)		XYZU+/V*-

33. **Convert the following infix expression to its equivalent postfix expression showing stack contents for the conversion: A+B*(C-D)/E**

Ans.

Symbol scanned from infix	Stack status (bold letter shows the top of the stack)	Postfix expression
	(	
(	**((**	
A	**((**	A
+	**((+**	A
B	**((+**	AB
)	**(**	AB+
*	**(***	AB+
(	**(*(**	AB+
C	**(*(**	AB+C
-	**(*(-**	AB+C
D	**(*(-**	AB+CD
)	**(***	AB+CD-
/	**(/**	AB+CD-*
E	**(/**	AB+CD-*E
)		AB+CD-*E/

Answer: Postfix expression of **(A+B)*(C-D)/E** is **AB+CD-*E/**

LONG ANSWER QUESTIONS

1. **Write an algorithm that depending upon user's choice, either pushes or pops an element in a stack implemented as an array. (The elements are not shifted after a push or a pop).**

Ans.

Algorithm for user choice:

1. ch=0 // Initialize a variable for user choice
2. Read ch //Enter, user choice 1 for push and 2 for pop
3. If(ch == 1) then /* if top is at end of stack-array */
4. call push function
5. Else
6. call pop function
7. End

Algorithm for pushing in stack-array:

/* Assuming that array-stack can hold maximum N elements*/

1. top=-1 /* Initialize stack with invalid subscript value to show that it is empty. Legal subscript values are 0..... (N-1) */
2. Read Item //Item, which is to be pushed
3. If(top== N-1) then /* if top is at end of stack-array */
4. { print "Overflow !!"
   }
5. Else
6. { top = top+1 /* Increment top to move to next empty position to hold new item*/
7. Stack[top]=Item }
8. END.

Algorithm for popping in stack-array:

/* Firstly, check for underflow condition */

1. If top==-1 then
   {
2. Print "Underflow!!"
3. Exit from program
   }
4. Else

	{
	5. print stack(top)
	6. top=top-1 /* top moved, so that the top most element becomes inaccessible which is a sort of a deletion */
	}
	7. END

2.	**A line of text is read from the input terminal into a stack. Write an algorithm to output the string in the reverse order, each character appearing twice (e.g., the string a b c d e should be changed to ee dd cc bb aa).**
Ans.	1. Ptr=top 2. While (ptr <> NULL or ptr <> 0) do steps 3 through 5. 3. { print ptr -> INFO, ptr -> INFO // print twice 4. Pop (stack, ITEM) 5. Ptr=ptr-1 }
3.	**Write an algorithm that depending upon user's choice, either pushes or pops an elements in a stack implemented as an array. (The elements are shifted towards right to combine all free spaces in the left)**

Ans.	**Algorithm for user choice:**
	1. ch=0 // Initialize a variable for user choice
	2. Read ch //Enter, user choice 1 for push and 2 for pop
	3. If(ch == 1) then /* if top is at end of stack-array */
	4. call push function
	5. Else
	6. call pop function
	7. End
	Algorithm for pushing in stack-array:
	
	/* Assuming that array-stack can hold maximum N elements*/
	1. top=-1 /* Initialize stack with invalid subscript value to show that it is empty. Legal subscript values are 0..... (N-1) */
	2. Read Item //Item, which is to be pushed
	3. If(top== N-1) then /* if top is at end of stack-array */
	4. { print "Overflow !!"
	}
	5. Else
	6. { top = top+1 /* Increment top to move to next empty position to hold new item*/
	7. Stack[top]=Item }
	8. END.
	Algorithm for popping in stack-array:
	
	/* Firstly, check for underflow condition */
	1. If top==-1 then
	{
	2. Print "Underflow!!"
	3. Exit from program
	}
	4. Else

	{
	5. print stack(top)
	6. top=top-1 /* top moved, so that the top most element becomes inaccessible which is a sort of a deletion */
	}
	7. END
	
	Note:
	We have to shift elements to the right for insertion and left for removal of an element. We face the same problem while implementing the list with the use of the array. If we push and pop the elements from the start of the array for stack implementation, this problem will arise. In case of push, we have to shift the stack elements to the right. However, in case of pop, after removing the element, we have to shift the elements of stack that are in the array to the left. If we push the element at the end of the array, there is no need to shift any element. Similarly as the pop method removes the last element of the stack which is at the end of the array, no element is shifted. To insert and remove elements at the end of the array we need not to shift its elements. Best case for insert and delete is at the end of the array where there is no need to shift any element. We should implement push() and pop() by inserting and deleting at the end of an array.

4.	**Write an algorithm that either pushes or pops an element in a linked stack, depending upon user's choice.**
Ans.	**Algorithm for user choice:** 1. ch=0 // Initialize a variable for user choice 2. Read ch //Enter, user choice 1 for push and 2 for pop 3. If(ch == 1) then /* if top is at end of stack-array */ 4. call push function 5. Else 6. call pop function 7. End **Algorithm for pushing in Linked-Stack** **Step-1:** If the Stack is empty go to step-2 or else go to step-3 **Step-2:** Create the new element and make your "stack" and "top" pointers point to it and quit. **Step-3:** Create the new element and make the last (top most) element of the stack to point to it **Step-4:** Make that new element your TOP most element by making the "top" pointer point to it. **Algorithm for popping in Linked-Stack:** **Step-1:** If the Stack is empty then give an alert message "Stack Underflow" and quit; or else proceed **Step-2:** If there is only one element left go to step-3 or else step-4 **Step-3:** Free that element and make the "stack", "top" and "bottom" pointers point to NULL and quit **Step-4:** Make "target" point to just one element before the TOP; free the TOP most element; make "target" as your TOP most element
5.	**Convert the expression (A+B)/(C-D) into postfix expression and then evaluate it for A=10, B=20, C=15, D=5. Display the stack status after each operation.**
Ans.	(A+B)/(C-D) Equivalent postfix expression: AB+CD-/ Evaluation of expression: Putting A=10, B=20, C=15 and D=5 into expression =AB+CD-/ =10 20 + 15 5 - / =[10+20] / [15 – 5] =30/10 =3
6.	**Write an algorithm to insert or delete an element from a queue (implemented as an array) depending upon user's choice. The elements are not shifted after insertion or deletion.**

Ans.

Algorithm for user choice:

```
1. ch=0                  // Initialize a variable for user choice
2. Read ch               //Enter, user choice 1 for push and 2 for pop
3. If(ch == 1) then      /* if top is at end of Array-Queue */
4. call insert function
5. Else
6. call delete function
7. End
```

Algorithm for inserting in Array-Queue:

```
1. If rear = NULL Then
  {
2.        rear=front=0
3.        QUEUE[0]=ITEM
  }
4. Else If rear= N-1 then
5.        Print "QUEUE FULL, OVERFLOW!!"
6. Else
  {
7.        QUEUE[rear+1]=ITEM
8.        rear=rear+1
  }
9. END.
```

Algorithm for deleting in Array-Queue:

```
1. If front==NULL Then
2. Print "Queue Empty"
3. Else
  {
4.   ITEM=QUEUE[front]
5.   If front=rear Then
     {
6.          front=rear=NULL
     }
7.   Else
8.   front = front + 1
  }                  /* end of step 3*/
9. END
```

7. **Write an algorithm to insert or delete an element from a queue (implemented as an array) depending upon user's choice. The elements are shifted towards left to combine all free space in the right.**

Ans.

Algorithm for user choice:

```
1. ch=0                  // Initialize a variable for user choice
2. Read ch               //Enter, user choice 1 for push and 2 for pop
3. If(ch == 1) then      /* if top is at end of Array-Queue */
4. call insert function
5. Else
6. call delete function
7. End
```

Algorithm for inserting in Array-Queue:

```
1. If rear = NULL Then
  {
2.        rear=front=0
3.        QUEUE[0]=ITEM
```

```
  }
4. Else If rear= N-1 then
5.        Print "QUEUE FULL, OVERFLOW!!"
6. Else
  {
7.        QUEUE[rear+1]=ITEM
8.        rear=rear+1
  }
9. END.
```

Algorithm for deleting in Array-Queue:

```
1. If front==NULL Then
2. Print "Queue Empty"
3. Else
  {
4.   ITEM=QUEUE[front]
5.   If front=rear Then
    {
6.          front=rear=NULL
     }
7.   Else
8.   front = front + 1
  }                  /* end of step 3*/
9. END
```

8. **Write an algorithm to insert or delete an element from a linked queue depending upon user's choice.**

Ans. **Algorithm for user choice:**

```
1. ch=0                    // Initialize a variable for user choice
2. Read ch                 //Enter, user choice 1 for push and 2 for pop
3. If(ch == 1) then        /* if top is at end of Array-Queue */
4. call insert function
5. Else
6. call delete function
7. End
```

Algorithm for inserting in Linked-Queue:

```
/* Allocate space for ITEM to be inserted */
1. NEWPTR = new node
2. NEWPTR -> INFO = ITEM; NEWPTR -> LINK=NULL
/* Insert in the queue */
3. If rear = NULL Then
  {
4.        front = NEWPTR
5.        rear = NEWPTR
6. else
  {
7.        rear -> LINK = NEWPTR
8.        rear=NEWPTR
  }
9. END.
```

Algorithm for deleting in Linked-Queue:

```
1. If front==NULL Then
```

	2. Print "Queue Empty"
	3. Else
	{
	4. ITEM=front->INFO
	5. If front=rear Then
	{
	6. front=rear=NULL
	}
	7. Else
	8. front = front + 1
	}
	9. END

9.	**Write an algorithm to implement input-restricted deque. (both the operations i.e., insertions and deletions should be taken care of).**
Ans.	**Algorithm to add an element into DeQueue :** Assumptions: pointer f,r and initial values are -1,-1 Q[] is an array max represent the size of a queue **enq_front** step1. Start step2. Check the queue is full or not as if (f <> step3. If false update the pointer f as f= f-1 step4. Insert the element at pointer f as Q[f] = element step5. Stop **enq_back** step1. Start step2. Check the queue is full or not as if (r == max-1) if yes queue is full step3. If false update the pointer r as r= r+1 step4. Insert the element at pointer r as Q[r] = element step5. Stop **Algorithm to delete an element from the DeQueue** **deq_front** step1. Start step2. Check the queue is empty or not as if (f == r) if yes queue is empty step3. If false update pointer f as f = f+1 and delete element at position f as element = Q[f] step4. If (f== r) reset pointer f and r as f=r=-1 step5. Stop **deq_back** step1. Start step2. Check the queue is empty or not as if (f == r) if yes queue is empty step3. If false delete element at position r as element = Q[r] step4. Update pointer r as r = r-1 step5. If (f == r) reset pointer f and r as f = r= -1 step6. Stop
10.	**Write algorithms to (i) insert an element in a circular queue and (ii) delete an element from a circular queue.**
Ans.	Here, CQueue is a circular queue where to store data. Rear represents the location in which the data element is to be inserted and Front represents the location from which the data element is to be removed. Here N is the maximum size of CQueue and finally, Item is the new item to be added. Initailly Rear = 0 and Front = 0. 1. If Front = 0 and Rear = 0 then Set Front := 1 and go to step 4.

2. If Front =1 and Rear = N or Front = Rear + 1
   then Print: "Circular Queue Overflow" and Return.
3. If Rear = N then Set Rear := 1 and go to step 5.
4. Set Rear := Rear + 1
5. Set CQueue [Rear] := Item.
6. Return

Here, CQueue is the place where data are stored. Rear represents the location in which the data element is to be inserted and Front represents the location from which the data element is to be removed. Front element is assigned to Item. Initially, Front = 1.

1. If Front = 0 then
   Print: "Circular Queue Underflow" and Return. /*..Delete without Insertion
2. Set Item := CQueue [Front]
3. If Front = N then Set Front = 1 and Return.
4. If Front = Rear then Set Front = 0 and Rear = 0 and Return.
5. Set Front := Front + 1
6. Return.

11. **Each node of a STACK contains the following information, in addition to pointer field:**
(ii) Pin code of city, (ii) Name of city
Give the structure of node for the linked STACK in question.
TOP is a pointer that points to the topmost node of the STACK. Write the following functions:
(i) PUSH ( ) – To push a node into the STACK, which is allocated dynamically.
(ii) POP () – to remove a node from the STACK, and release the memory.

Ans.

```
struct City
{
      long Cpin ;
      char CName[20] ;
      City *Next ;
} ;
class Stack
{
      City *Top;
       public:
      Stack( ) { Top = NULL; }
      void Push( );
      void Pop( );
      void Display( );
};
void Stack::PUSH( )
{
      City *Temp;
      Temp=new City;
      if(Temp= =NULL)
      {
            cout<<"\nNo memory to create the node…";
            exit(1);
      }
      cout<<"\nEnter the City Pin Code to be inserted: ";
      cin>>Temp → Cpin;
      cout<<"\nEnter the City Name to be inserted: ";
      gets(Temp → CName);
      Temp → Next=Top;
      Top=Temp;
}
void Stack::POP( )
{
```

```
      City *Temp;
      if( Top= = NULL)
            cout<<"Stack Underflow…";
      else
      {
            cout<<"\nThe City Pin Code for the element to delete: "<<Top→Cpin;
            cout<<"\nThe City name of the element to delete: "<<Top→CName;
            Temp=Top;
            Top=Top → Next;
            Delete Temp;
      }
}
```

12. **Write a function in C++ to Delete an element into a dynamically allocated Queue where each node contains a real number as data. Assume the following definition of MYNODE for the same.**

```
struct MYNODE
{
     float NUM;
     MYNODE * Link;
};
```

Ans.

```
struct MYNODE
{
      float NUM;
      MYNODE *Link;
};
class Queue
{
      MYNODE *front,*rear;
    public:
      Queue( )
      {
            front=rear=NULL;
      }
      void Insert( );
      void Delete( );
      void Display( );
};
void Queue::Delete( )
{
      MYNODE *temp;
      if(front= = NULL)
            cout<<"Queue Underflow";
      else
      {
            cout<<"\nThe content of the element to delete:"<<front → NUM;
            temp=front;
            front=front → Link;
            delete temp;
      }
}
```

13.

```
class queue
{
      int data[10] ;
      int front, rear ;
  public :
      queue( ) { front = - 1 ; rear = - 1 ; }
```

void add( ) ; //to add an element into the queue
void remove( ) ; //to remove an element from the queue
void Delete(int ITEM( ) ; //to delete all elements which are equal to ITEM
};

Complete the class with all function definitions for a circular array Queue. Use another queue to transfer data temporarily.

Ans.

```
class queue
{
            int data[10] ;
            int front, rear ;
    public :
            queue( ) { front = - 1 ; rear = - 1 ; }
             void add( ) ;
             void remove( ) ;
              void Delete(int ITEM( ) ;
} ;
void queue::add()
{
     if((front==0 && rear==9)||(rear==front-1))
     {
          cout<<"Over flow";
          return;
     }
     int X;
     cout<<"Enter the data:";
     cin>>X;
     if(rear==-1)
     {
          front=0;
          rear=0;
     }
     else if(rear==9)
          rear=0;
     else
          rear++;
     data[rear]=X;
}
void queue::remove()
{
     if(front==-1)
     {
          cout<<"Underflow";
          return;
     }
     int X;
     X=data[front];
     if(front==rear)
     {    front=-1;
          rear=-1;
     }
     else if(front==9)
          front=0;
     else
          front++;
     cout<<X<<"Removed";
```

```
}
void queue::DElete(int ITEM)
{
      if(front==-1)
      {
            cout<<"Underflow";
            return;
      }
      queue t;
      t.front=0;
      while(1)   //will repeat endless untill encounters a break statement
      {
            if(data[front]!=ITEM)
            {     t.rear++;
                  t.data[t.rear]=data[front];
            }
            if(front==rear)
                  break;
            front++;
            if(front==10)
                  front=0;
      }
      //copy temp data to current data
      front=0;
      rear=-1;
      while(t.frot<=t.rear)
      {
            rear++;
            data[rear]=t.data[t.front];
            t.front++;
      }
}
```

14.

Introduction

```
class stack
{
      int data[10] :
      int top ;
public :
     stack( )
          { top = - 1; }
     void push( ) ;          //to push an element into the stack
     void pop( ) ;           //to pop an element from the stack
    void Delete(int ITEM) ;          //To delete all elements which are equal to ITEM.
};
```

Complete the class with all function definitions. Use another stack to transfer data temporarily.

Ans.

```
class stack
{
            int data[10] :
            int top ;
public :
          stack( )
                    { top = - 1; }
          void push( ) ;
          void pop( ) ;
          void Delete(int ITEM);
```

```
        void display()
        {
            for(int i=top;i>=0;i--)
                 cout<<data[i]<<"\n";
        }
} ;
void stack::push()
{
      if(top==9)
      {     cout<<"Overflow";
            return;
      }
      int X;
      cout<<"Enter the data:";
      cin>>X;
      top++;
      data[top]=X;
}
void stack::pop()
{
      if(top==-1)
      {     cout<<"Uderflow";
            return;
      }
      int X;
      X=data[top];
      top--;
      cout<<X<<"Removed";
}
void stack::Delete(int ITEM)
{
      stack t;
      if(top==-1)
      {
            cout<"underflow";
            return;
      }
      while(top>=0)
      {
            if(data[top]!=ITEM)
            {
                  top++;
                  t.data[t.top]=data[top];
            }
            top--;
      }
      while(t.top>=0)       //copy the emp data to current data
      {
            top++;
            data[top]=t.data[t.top];
            t.top--;
      }
}
```

15. **Define functions stackpush() to insert nodes and stackpop() to delete nodes, for a linked list implemented stack having the following structure for each node:**
struct node

```
{
     char name[20];
     int age;
     node *Link;
};
```

Ans.

```
void stack::stackpush()
{
      node *nptr;
      nptr=new node;
      nptr->link=NULL;
      cout<<"Enter name for new node:";
      gets(nptr->name);
      cout<<"Enter age for new node:";
      cin>>nptr->age;
      if(top==NULL)
            top=nptr;
      else
      {
            nptr->link=top;
                  top=nptr;
      }
}
void stack::stackpop()
{
      if(top==NULL)
            cout<<"Underflow";
      else
      {
            cout<<"Element being popped is \n";
            cout<<top->name<<":"<<top->age<<endl;
            node *ptr;
            ptr=top;
            top=top->link;
            delete ptr;
      }
}
```

16. **Define member functions queins( ) to insert nodes and quedel ( ) to delete nodes of the linked list implemented class queue, where each node has the following structure:**

```
struct node
{
   char name[20] ;
   int age ;
   node *Link ;
};
class queue
{
   node *rear, *front ;
 public :
   queue( ) { rear = NULL;front = NULL} ;
    void queins( ) ;
    void quedel( ) ;
};
```

Ans.

```
void queue::queins( )
{
      node *ptr;
```

```
      ptr=new node;
      if(ptr= = NULL)
      {
            cout<<"\nNo memory to create a new node….";
            exit(1);
      }
      cout<<"\nEnter the name….";
      gets(ptr → name);
      cout<<"\nEnter the age…";
      cin>>ptr → age;
      ptr → Link=NULL;
      if(rear= = NULL)
            front=rear=ptr;
      else
      {
            rear → Link=ptr;
            rear=ptr;
      }
}
void queue::quedel( )
{
      node *temp;
            if(front= = NULL)
            cout<<"Queue Underflow";
      else
      {
            cout<<"\nThe name of the element to delete: "<<front → name;
            cout<<"\nThe age of the element to delete: "<<front → age;
            temp=front;
            front=front → Link;
            delete temp;
      }
}
```

17. **Write a function in C++ to Delete an element from a dynamically allocated Queue where each node contains a real number as data. Assume the following definition of MYNODE for the same.**

```
struct MYNODE
{
      float NUM;
      MYNODE * Link;
};
```

Ans.

```
struct MYNODE
{
      float NUM;
      MYNODE *Link;
};
class Queue
{
      MYNODE *front,*rear;
    public:
      Queue( )
      {
            front=rear=NULL;
      }
      void Insert( );
      void Delete( );
      void Display( );
```

```
};
void Queue::Delete( )
{
      MYNODE *temp;
      if(front= = NULL)
            cout<<"Queue Underflow";
      else
      {
            cout<<"\nThe content of the element to delete:"<<front → NUM;
            temp=front;
            front=front → Link;
            delete temp;
      }
}
```

18. **Write a function QUEINS ( ) in C++ to insert an element in a dynamically allocated Queue containing nodes of the following given structure:**

```
struct Node
{
   int PId; // Product Id
   char Pname[20];
   NODE *Next;
};
```

Ans.

```
Class Queue
{
      Node *Front, *Rear;
         public:
      QUEUE( ) // Constructor to initialize Front and Rear
      {
            Front = NULL;
            Rear = NULL;
      }
      void QUEINS( ); // Function to insert a node
      void QUEDEL( ); // Function to delete a node
      void QUEDISP( ); // Function to display nodes
      ~Queue( ); //Destructor to delete all nodes
};
void Queue::QUEINS( )
{
      Node *Temp;
      Temp = new Node;
      cin>> Temp->PId;
      gets(Temp->Pname);
      Temp->Next=NULL;
      if (Rear = = NULL)
      {
            Front = Temp;
            Rear = Temp;
      }
      else
      {
            Rear->Next = Temp;
            Rear = Temp;
      }
}
```

19. **Write a function QUEDEL ( ) in C++ to display and delete an element in a dynamically allocated Queue containing nodes of the following given structure:**

	struct Node **{** **int Itemno;** **char Itemname[20];** **NODE *Link;** **};**

Ans.	struct Node
	{
	int itemno;
	char Itemname[20] ;
	Node *Link ;
	} ;
	class Queue
	{ Node *front,*rear;
	public:
	Queue( )
	{
	front=rear=NULL; }
	void Insert( );
	void Delete( );
	void Display( );
	};
	void Queue::Delete( )
	{
	Node *Temp;
	if(front= =NULL)
	cout<<"Queue Underflow. No element to delete";
	else
	{
	cout<<"\n The item number for the element to delete.<<front->CNo;
	cout<<"\n The item name for the element to delete"<<front->CName;
	Temp=front;
	front = front->Link;
	delete Temp;
	}
	}