Assignment - 1

1. Algorithm to find sum of digit of an array
   Step 1: Start
   Step 2: Initialize arr[] = {1, 2, 3, 4, 5}
   Step 3: set sum = 0
   step 4: length = size of (arr) / size of (arr[0])
   Step 5: set i = 0 i < n. Repeat step 6 & 7 until i < length
   Step 6: Sum = Sum + arr[i]
   Step 7: i = i + 1
   Step 8: Print "sum of all elements of an array" by assigning sum.
   step 9: Return 0
   step 10: End

2. Algorithm to find largest & smallest element in array.
   step 1: Start
   Step 2: Input the array elements
   Step 3: Initialize small = large = arr[0]
   Step 4: Repeat from i = 2 to n
   step 5: if (arr[i] > large)
   step 6: large = arr[i]
   step 7: if (arr[i] < small)
   Step 8: small = arr[i]
   Step 9: Print small & large
   step 10: Return 0
   step 11: End

3) Algorithm To print reserve of an array.

Step 1: start
Step 2: Initialize arr[] = {1, 2, 3, 4, 5}
step 3: length = size of (arr) / size of arr[0]
step 4: Print "original array".
Step 5: Repeat step 6 & 7 until i < length
Step 6: Print arr[i]
Step 7: i = i+1
step 8: Print new line
Step 9: Print "Array in reverse order"
Step 10: Set i = length - 1. Repeat step 11 & 12 until i >= 0

Step 11: Print a[i]
Step 12: i = i-1
Step 13: Return 0
Step 14: End

4. a. K + L - M * N + (O ^ P) * W / U / V * T + Q

Input	stack	Post fix Exp	Rank
K	K	K	0
+	+	K	1
L	+ L	K	1
-	-	K L +	1
M	- M	K L +	1
*	- *	K L + M	2
N	- * N	K L + M	2
+	+	K L + M N * -	1
(	+ (	K L + M N * -	1
O	+ (O	K L + M N * -	1
^	+ (^	K L + M N * - O	2
P	+ (^ P	K L + M N * - O	2

)	+	KL+MN*-OP^	2
*	+ *	KL+MN*-OP^	2
W	+ * W	KL+MN*-OP^W*	2
/	+ /	KL+MN*-OP^W*U/	2
V	+ / V	KL+MN*-OP^W*U/V/	2
*	+ *	KL+MN*-OP^W*U/V/	2
+	+ * +	KL+MN*-OP^W*U/V/	2
+	+	KL+MN*-OP^W*U/V/T*+	1
Q	+ Q	KL+MN*-OP^W*U/V/T*+Q+	1
#	#	KL+MN*-OP^W*U/V/T*+Q+	1

Since rank = 1 (Expression is valid.

Postfix expression : KL+MN*-OP^W*U/V/T*+Q+

b. A+B*-F+A*E

Reverse expression : E*A+E-*B+A

Input	Stack	Output	Rank
E		#	
*	* E		
A	* A	E	1
+	+	EA*	1
E	+ E	EA*	1
-	-	EA*F+	1
*	-*	EA*F+	1
B	-*B	EA*F+	1
+	+	EA*F+B*-	0
A	+A	EA*F+B*-	0
		EA*F+B*-A+	0

K. J. Somaiya Institute of Technology, Sion (E), Mumbai-400022.

Output is: EA^*F+B^*-A+

since rank = 0, expression is not valid.

Conversion from infix to postfix

Input	stack	Output	Rank
A	#	A	
+	+	A	1
B	+B	AB	1
*	+*	AB	2
-	-	AB*+	0
F	-F	AB*+	0
+	+	AB*+F-	0
A	+A	AB*+F-A	0
*	+*	AB*+F-A	1
E	+*E	AB*+F-A	1
#	#	AB*+F-AE*+	0

Q4] 0 Post fix expression : AB^*+F-AE^*+

since rank = 0, expression is not valid

Q5] a) $ABC/-AK/L-*$

Input	stack	Infix Exp	Rank
A	A	#	0
B	AB	#	0
C	ABC	#	0
/	A(B/C)	#	0
-	(A-(B/C))	#	0
A	(A-(B/C))A	#	0
K	(A-(B/C))AK	#	0
/	(A-(B/C))/(A/K)	#	0
L	(A-(B/C))/((A/K)-L)	#	0

-	(A-(B/C))*((A/K)-L)	#	0
*	(A-(B/C))*((A/K)-L)	#	0
#	#	(A-(B/C))*((A-K)-L)	1

since rank = 1, Expression is valid

Converting to prefix:

Reversing expression :)L-)K-A((*))C/B(-AC

input	stack	output	Rank
)	)	#	0
L	)L	#	0
-	)-	L	1
)	)-)	L	1
K	)-)K	LK	1
/	)-)/	LK	2
A	)-)/A	LK	2
(	)-	LKA/	2
(	#	LKA/-	1
*	#*	LKA/-	1
)	#*)	LKA/-	1
)	#*))	LKA/-	1
C	#*))C	LKA/-	1
/	#*))/	LKA/-C	2
B	#*))/B	LKA/-C	2
C	#*)	LKA/-CB/	2
-	#*)-	LKA/-CB/	2
A	#*)-A	LKA/-CB/A-	2
C	#*	LKA/-CB/A-	2

Department of: - /Class /Sem. /Sub. /Academic Year:-

A L K A I - C O I A - P 1

since rank = 1, expression is valid

b. Postfix to Infix

Input	Stack	Output	Rank
A	A	#	0
B	A B	#	0
C	A B C	#	0
D	A B C D	#	0
^	A B C (C ^ D)	#	0
E	A B C (C ^ D) E	#	0
-	A B ((C ^ D - E)	#	0
*	A C D * ((C ^ D - E))	#	0
+	A + (B * ((C ^ D) - E))	#	0
#	#	A + (B * ((C ^ D) E))	1

since rank = 1, expression is valid
Then to convert this expression to prefix
Reversing expression :-))) E -) D ^ ((C * B (+ A (

Input	stack	output	Rank
A	A	#	0
B	A B	#	0
C	A B C	#	0
*	A (B * C)	#	0
+	(A + (B * C))	#	0
D	(A + (B * C)) D	#	0
+	((A + (B * C)) + D)	#	0
E	((A + (B * C)) + D) E	#	0

F	((A+(B*C)+D)EF	#	0
#	#	((A+(B*C)+D)EF	3

Infix Expression is ((A+(B*C)+D)EF
Rank = 3, thus, expression is not valid.

Converting this into prefix:
Reversing the expression FE)D+))C*B(+A((

Input	stack	output	Rank
F	F	#	0
E	E	F	1
)	)	FE	2
D	)D	FE	2
+	)+	FED	3
)	)+)	FED	3
)	)+))	FED	3
C	)+))C	FED	3
*	)+))C*	FEDC	4
B	)+))*B	FEDC	4
(	)+)	FEDCB*	4
+	)+)+	FEDCB*	4
A	)+)+A	FEDCB*	4
(	)+	FEDCB*A+	4
(	#	FEDCB*A++	3

∴ Prefix expression is ++A*BCDEF
Rank = 3, thus expression is invalid.

Q6 a.) Reversing expression : HG+FE//DCBA^K-P

Input	Stack	Output	Rank
H	H	#	0
G	HG	#	0
+	(G+H)	#	0
F	(G+H)F	#	0
E	(G+H)FE	#	0
/	(G+H)(E/F)	#	0
/	((E/F)/(G+H))	#	0
D	((E/F)/(G+H))D	#	0
C	((E/F)/(G+H))DC	#	0
B	((E/F)/(G+H))DCB	#	0
A	((E/F)/(G+H))DCBA	#	0
^	((E/F)/(G+H))D(C^B)	#	0
*	((E/F)/(G+H))D((A^B)*C)	#	0
-	((E/F)/(G+H))D(((A^B)*C)-D)	#	0
+	(((((A^B)*C)-D)+((E/F)/(G+H)))	#	0
#	#	(((((A^B)*C)-D)+((E/F)/(G+H)))	1

Infix expression is (((((A^B)*C)-D)+((E/F)/(G+H)))
Since Rank = 1 expression is valid

Converting to postfix
(((((A^B)*C)-D)+((E/F)/(G+H)))

Input	Stack	Output	Rank
(	(	#	0
(	((	#	0
(	(((	#	0

K. J. Somaiya Institute of Technology, Sion (E), Mumbai-400022.

A	(((A	#	0
+	(((+	A	1
B	(((+B	A	1
)	((	AB+	1
*	((*	AB+	1
(	((*(	AB+	1
(	((*((	AB+	1
C	((*((C	AB+	1
+	((*((+	AB+C	2
D	((*((+D	AB+C	2
)	((*(+	AB+CD+	2
-	((*(-	AB+CD+	2
E	((*(-E	AB+CD+	2
)	((*	AB+CD+E-	2
)	(	AB+CD+E-*	1
*	(*	AB+CD+E-*	1
F	(*F	AB+CD+E-F*	1
)	#	AB+CD+E-F*	1

Post fix Expression is AB+CD+E-F*
Rank = 1, thus expression is valid.

a. a) 100 200 + 2 / 5 * 7 +

→ Rank = 1
thus this expression can be solved

Input	Stack
100	(100)
200	(100) (200)
+	(300) (2)

/	(300/2)
5	(150)(5)
*	(150 * 5)
7	(150 * 5)(7)
+	(750 + 7)
	(757

∴ 100 200 + 2/5 * 7 + = 757

b) 2 5 3 6 + * * 5/2 −

→ Rank = 1 + 1 + 1 + 1 + 1 − 1 − 1 − 1 + + − 1 + 1 − 1

Rank = 1

thus this expression can be solved

Input	Stack
2	(2)
5	(2)(5)
3	(2)(5)(3)
6	(2)(5)(3)(6)
+	(2)(5)(3+6)
*	(2)(5 * 9)
*	(2 * 45)
5	(90)(5)
/	(90/5)
2	(18)(2)
−	(18−2)
#	(16)

∴ 2 5 3 6 + * * 5/2 − = 16

7. 1. A deque is a list in which the elements can be inserted or deleted at either end.

2. However, no element can be added or deleted from the middle of the dequeue.

3. In a dequeue two pointers are maintained, LEFT and RIGHT, which point to either end of the dequeue.

4. Types of dequeue

(a) Input restricted dequeue - In this dequeue, insertions can be done at one of the ends, while deletions can be done from both ends.
(b) Output restricted dequeue - In this dequeue, deletions can be done only one end while insertions can be done at both ends.

5. Algorithm:-

(a) Insertion at Front

PROCEDURE DEQ_FINSERT(Q,F,R,N,Y)
1 [Check for availability]
if F=1
then write ('Cannot add element at front')
return
2. [Decrement front pointer]
F ← F-1
3 [Insert element]
Q[F] ← Y
return

(b) Insertion at Rear:

1. [check for availability]
   if R = N
   write ('Cannot add elements at rear')
   return
2. [Increment rear pointer]
   R ← R+1
3. [Insert element]
   Q[R] ← X
   return

(C) Deletion at front

Function Deq F Delete(Q, F, R, N, Y)

1. [check for availability]
   if F = N
   write ('Cannot delete value at front')
   return
2. [Delete element]
   Y ← Q[F]
3. [check Queue empty or increment front pointer]
   if F = R
   then F ← R ← 0
   else F ← F+1
4. [Return element]
   return (Y)

Applications of Dequeue:

1. Dequeues can be used to manage the order and priority of incoming tasks or request.
2. Dequeues can be used to store nodes and perform graph traversal algorithms operations like adding

or removing nodes from both ends of the dequeue
3. Dequeues can be used to check if a string or number is a palindrome.
4. Dequeues can be used to make a undo operations of a software application.

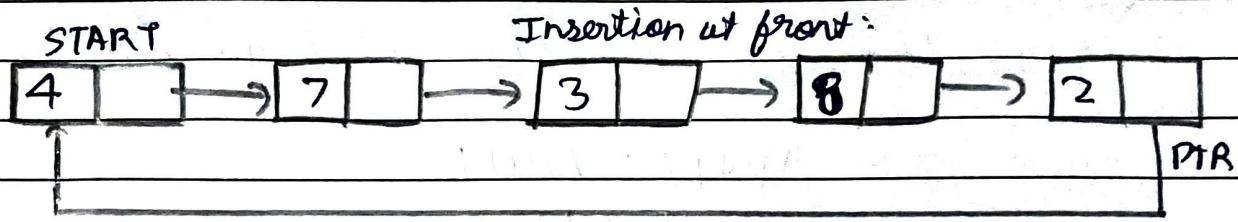
Q8 → 1. A linked list is a data structure that consists of of a sequence of nodes, each containing some data and a pointer to the next node.
2. A linked can be used to store and manipulate data in a student information management system.
3. Algorithm for insertion at beginning

Step 1: IF AVAIL = NULL
Write OVERFLOW
Go to step 7
[END OF IF]
Step 2: SET NEW_NODE = AVAIL
~~Step 3: SET NEW_NODE →~~
Step 3: SET AVAIL = AVAIL → NEXT
Step 4: SET NEW_NODE → DATA = VAL
Step 5: SET NEW_NODE → NEXT = START
Step 6: SET START = NEW_NODE
Step 7: EXIT

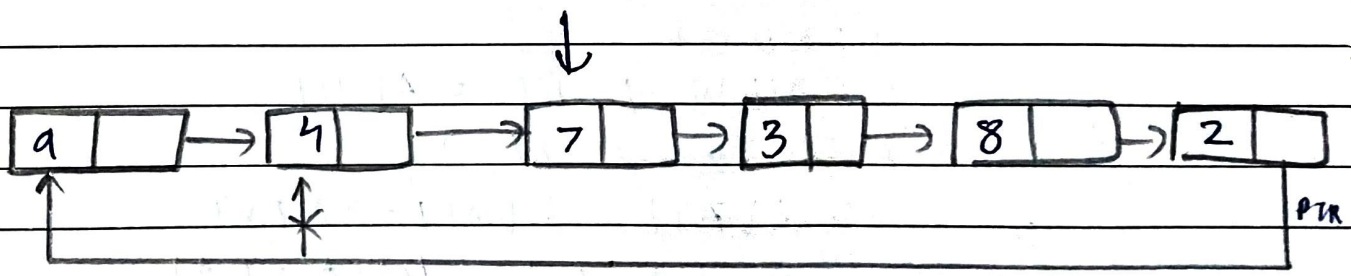
4. Algorithm for insertion at end
Step 1: IF AVAIL = NULL
Write OVERFLOW
GO TO step 10
[END OF IF]

Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL → NEXT
Step 4: SET NEW_NODE → DATA = VAL
Step 5: SET NEW_NODE → NEXT = NULL
Step 6: SET ~~START = NEW~~ NODE PTR = START
Step 7: Repeat step 9 while PTR → NEXT != NULL
Step 8: SET PTR = PTR → NEXT
[END OF LOOP]
Step 9: SET PTR → NEXT = NEW_NODE
Step 10: EXIT

Q 9.



Data to be inserted 9



Algorithm

Step 1: IF AVAIL = NULL
write Overflow
Go to step 1
[END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET AVAIL = AVAIL → NEXT
Step 4: SET NEW_NODE → DATA = VAL