ASSIGNMENT - 3

Q:1

(a) Write an algorithm to delete N^{th} element from front of circular queue.

Ans: deletion - circular - queue (queue, front, rear, val, size)

step 1 : if front = = -1
then print "underflow and exit";
step 2 : else
val = queue [front];
step 3 : if front = = rear;
then set front = rear = -1;
step 4 : if front = = size - 1
then set front = 0;
step 5 : else
front = front + 1;
step 6 : deleted value is "val";
step 7 : exit.

(b) Circular Queue is to be implemented using a array of 10 elements ..... queue is empty or not.

Ans: inserting - values - in - queue (queue [10], front, rear, val, size);

step 1 : if rear = = size - 1
then print "overflow and exit";
step 2 : if front = = rear = -1
then set front = rear = 0;
step 3 : else
set rear = rear + 1;
step 4 : set queue [rear] = val;
step 5 : exit

Q:2 (a) Translate Infix expression into its equivalent postfix exp" (using stack):

P: (A * (B + D) / E - F * (G + H / K))

symbols scanned	stack	Postfix exp"
(	(	
A	(	A
*	(*	A
(	(* (	A
B	(* (	AB
+	(* (+	AB
D	(* (+	ABD
)	(*	ABD +
/	(/	ABD + *
E	(/	ABD + * E
-	(-	ABD + * E /
F	(-	ABD + * E / F
*	(- *	ABD + * E / F
(	(- * (	ABD + * E / F
G	(- * (	ABD + * E / FG
+	(- * (+	ABD + * E / FG
H	(- * (+	ABD + * E / FGH
/	(- * (+ /	ABD + * E / FGH
K	(- * (+ /	ABD + * E / FGHK
)	(- *	ABD + * E / FGHK / +
)	-	ABD + * E / FGHK / + * -

* Postfix Expression is : ABD + * E / FGHK / + * -

(b) Consider the following arithmetic exp" P written in Postfix Notation.

P : 12, 7, 3, -, /, 2, 1, 5, +, *, +

Evaluate the Postfix expression.

symbols scanned	stack	Postfix Expression
12	-	12
7	-	12 7
3	-	12 7 3
-	-	12 7-3
/	-	12/4
2	-	3 2
1	-	3 2 1
5	-	3 2 1 5
+	-	3 2 1+5
*	-	3 2*6
+	-	3 + 12
		15

* Answer = 15

Q:3

(a) Consider the following queue of characters, where QUEUE is a circular array

FRONT : 2 QUEUE = _ , A , C , D , _ , _

REAR = 4

Describe the queue as the following operations take place:

a) F is added to the queue
b) two letters are deleted
c) K, L, M added to the queue
d) two letters are deleted
e) R added to the queue
f) two letters are deleted
g) S is added to the queue
h) two letters are deleted
i) one letter is deleted
j) one letter is deleted.

Ans:

	①	②	③	④	⑤	⑥	
	_	A	C	D	_	_	FR = 2 , RE = 4
a)	_	A	C	D	F	_	FR = 2 , RE = 5
b)	_	_	_	D	F	_	FR = 4 , RE = 5
c)	L	M	_	D	F	K	FR = 4 , RE = 2
d)	L	M	_	_	_	K	FR = 6 , RE = 2
e)	L	M	R	_	_	K	FR = 6 , RE = 3

f) &, M, R, –, –, #

g) –, M, R, S, –, –

h) –, –, –, S, –, –

i) –, –, –, –, –, –

j) the queue is empty. operation cannot be performed.

(b) write a program or algorithm to reverse a singly linked list.

Ans:

```
reversing_singlylinkedlist ()
{ node * prev = NULL, * next = NULL, *current = start;
  while (current != NULL)
    { next = current -> link;
      current -> link = prev;  prev = current;
      current = next;
    }
  start = prev;
}
```

Q:4

(a) Explain PUSH() and POP() operations used in stack with examples.

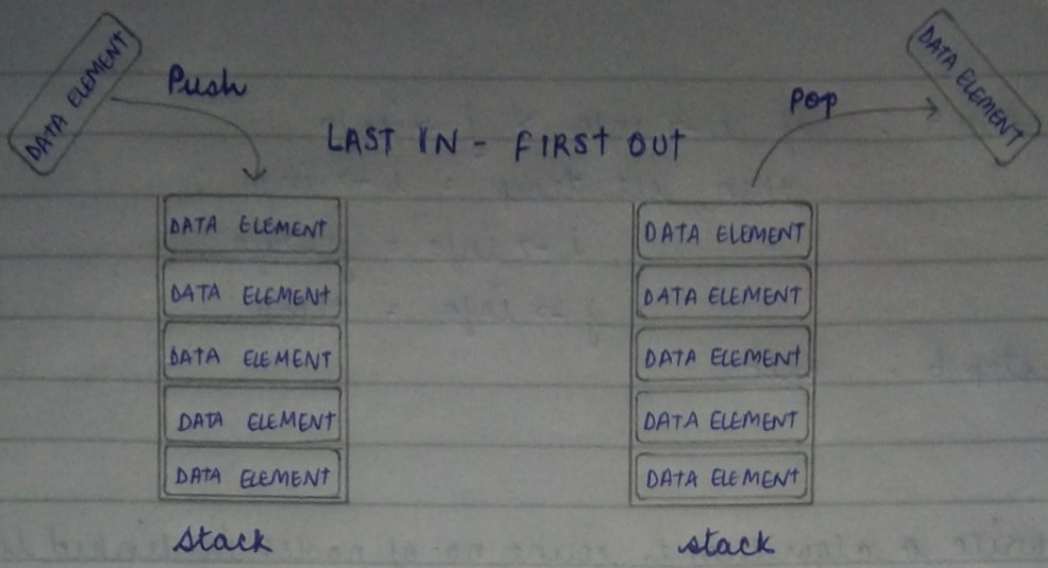
- Stack operations may involve initializing the stack, using it and then de-initializing it. Apart from these basic stuffs, a stack is used for the following two primary operations:

i) PUSH():

Push operation is the operation which is used to insert a new element in the stack. also Pushing (storing) an element on the stack.

ii) POP():

Pop operation is the operation which is used to delete a element from the stack. also Removing (accessing) an element from the stack.



(b) Write a pseudo code to delete a particular value from linked list.

Ans:

```
delete_particular_element()
{   node * ptr1 = NULL, * ptr2 = start ; int val;
    while (ptr21 = NULL)
      { if (ptr2 -> info == val)
          { flag = 1 ; break;
          }
        else
          { ptr1 = ptr2 ;
            ptr2 = ptr2 -> link ;
          }
      }
    ptr1 -> link = ptr2 -> link ;
    ptr1 = NULL ; ptr2 = NULL;
    free (ptr1) ; free (ptr2) ;
}
```

(c) Write a algorithm to sort a linked list.

sorting_linked_list (start)

step 1 : declare node *i , *j ;

step 2 : declare int temp ;

step 3 : Repeat step 4 for i = start to i -> link != NULL

step 4 : Repeat step 5 for j = i -> link to j != NULL

step 5 : if i -> info > j -> info
then set temp = i -> info ;
i -> info = j -> info ;
j -> info = temp ;

step 6 : exit

Q:5

(a) write a algorithm to count no. of nodes in linked list

counting_nodes (start)
step 1 : declare int count = 0 ;
step 2 : declare node * counter ;
step 3 : Repeat step 4 for counter = start to counter -> link != NULL
step 4 : write count ++ ;
step 5 : print " No. of Nodes are count + 1 " ;

(b) write an algorithm which removes 1st element of a list. Add it to the end of the linked list.

deleting_first_adding_at_last (start)
step 1 : declare node * delete , * ptr ;
step 2 : set delete = start
step 3 : set start = start -> link ;
step 4 : set ptr = start ;
step 5 : Repeat step 6 while ptr -> link != NULL
step 6 : set ptr = ptr -> link ;
step 7 : ptr -> link = delete ;
step 8 : exit ;