1. Define Dequeue.

2. Define Array. Write a note on Array Operations, representation and application of Arrays.

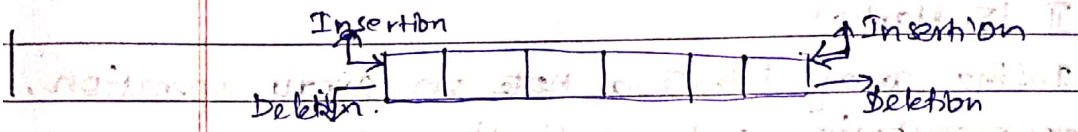
3. Define stack. Write the applications of stack.

4. Define Queue. Write the different types of Queue. Explain Circular Queue.

ANSWER KEY.

1. Define Dequeue. (5mark)

Dequeue is the linear list of elements in which the insertion and deletion operations are made at ~~both~~ either front or rear. A dequeue is more general in stack and linear queue and it ~~is~~ is in sort of FILFLO (First In First Out Last In Last Out).



There are two types of Dequeue.

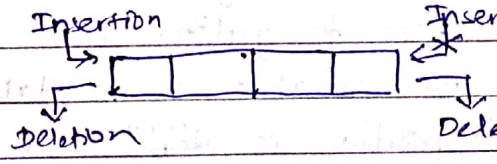
DEQUEUE

Input restricted Queue

Output Restricted Queue.

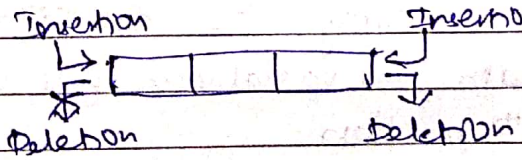
Input Restricted Dequeue.

It is a type of dequeue in which the input or insertion of elements are restricted on Rear end in a given queue. However, the deletion operation is done at either ends.



Output Restricted Dequeue:

The dequeue at which deletion or output of the elements to front end are restricted but the insertion can be done at (both) either ends.



The manipulation of Dequeue is done using two variables, as LEFT and RIGHT, and their behavior is given below.

	LEFT :	RIGHT :
Insertion.	Decremented by 1 mod n	Incremented by 1 mod n
Deletion.	Incremented by 1 mod n	Decrement by 1 mod n.

II. 15 Marks:

2. Define array. Write a note on array operation, representation and application of array.

Definition:

Array is an abstract ~~collection~~ data type at which the collection of set of elements are of homogeneous data type. It can be single dimentional, two dimentional and, in general multidimentional.

Premitive operations of array:

1. Retrival / Extraction:

Extraction or retrival of elements removes an element from the given set of elements or ~~set~~ array. It refers to getting a value from array and stored in an variable as x = data[i], where the data at i^{th} position is ~~stored~~ extracted and stored into variable x.

2. Storing:

Storing of elements refer to insertion of a given data into an array, as data[i] = x. This denotes that the data on variable x gets stored to the array of i^{th} position.

Representation:

↳ Array can be of single dimentional, 2 dimentional or in general multidimentional. The representation is given below.

(i) Single dimentional:

→ It is a simple form of array which denoted by ~~data~~ data type variable[dimension];

→ Example; int a[50], This represents the variable a is of integers type that can store 50 elements.

Let an array be a[1:n] where n → no. of elements. Let u be the upper index and l be lower index then,

no. of elements = (u-l+1)

The address of i^{th} element = $\alpha + (i-l)$

where α = base address.

(ii) Two dimentional:

→ It is a type of array at which represents in two sets, one as row and other as column.

→ It is also refered to as matrix.

→ However, the memory representation of elements in 2D array as to be single dimentional, so it makes 2 approaches for respresentation.

1. Row major representation:

The Row of matrix gets first priority.

2. Column major representation:

The column of matrix gets first priority.

Let a[1:n, 1:m] be 2D array of n row and m column.

So no. of elements is = $(u_1-l_1+1)(u_2-l_2+1)$

where u_1 & u_2

where u_1, u_2 → Upper bound of Row & column
l_1, l_2 → lower bound of Row & column.

Address of $(i,j)^{th}$ element = $\alpha + (i-l_1)(u_2 - b_2 + 1) + (j - l_2)$
where α = base address.

Application of array:

1. Sparce Matrix:

Sparce matrix is a matrix of 0's as dominating element. It is applied as 2 dimentional array, where the other matrix of representation is formed.

Example:

Let A be the matrix given below and B be the representation matrix. In representation matrix form A,

- A first row denotes the statistics like no. of non-zero rows, columns and also no. of non-zero elements.
- The rest will be the exact position of row, column and a non-zero value to be placed in that row and column.

$$A = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 4 & 4 & 3 \\ 1 & 1 & 2 \\ 2 & 3 & 3 \\ 3 & 2 & 1 \end{bmatrix}$$

2. Ordered list:

One of the easier and most useful object data in computer science is ordered list. It is a list of elements of ordered manner. It can do insertion, deletion and retrival operation.

If $a_1, a_4, a_7, a_9, a_{10}$ is a given list,

(i) Insertion	Insert a_5	a_1, a_4, a_5, a_7, a_9.
(ii) Delete	Delete a_7	a_1, a_4, a_5, a_9.
(iii) Retrival	Update a_6	a_1, a_4, a_5, a_6, a_9.

3. Define stack. Write down the application of stack.

Definition:

A linear list of elements at which insertion and deletion ~~of operation~~ of element is done at one of the ends is called stack. It makes insertion of element on the rear position and the deletion is also done to that same position. That is why, it makes an approach of FIFO (First In First Out).

Applications of Stack:

Stack is applied in many real life situation like recursion, towers of hanoii, ~~performance~~ well nesting of parameter, syntax checking, convertion of infix to postfix expression, ~~&~~ evaluation of postfix etc;

(i) INFIX, POSTFIX, PREFIX OPERATIONS:

An expression can be of 3 forms, such as ~~postfix~~, infix and postfix. These 3 forms differs in the positions of operand and operator. The operator and operand were combined to form a ~~the~~ operator result.

→ The expression which the operator comes before ~~after~~ the operand, it is called prefix expression. Eg. +AB

→ The expression which the operator comes in between ~~before~~ the operand, it is called infix expression. Eg: A+B.

→ The expression which the operator comes after. ~~before~~ The operand, it is called post fix expression. Eg: ABC+-

→ INFIX TO POSTFIX CONVERTION:

One of the application of stack is Infix to postfix convertion. To convert infix to postfix expression, we need to find the notation of operators which follows the priority as,

(i) Paranthesis ('(')
(ii) Exponential (↑)
(iii) Division / Multiplication. (÷, *)
(iv) Adelition / Subtraction (+, -)

Algorithm:

~~① Push "(" onto stack, and add ")" to the end of P.~~
~~② Scan~~ x = get next char(E).
while x ≠ $ do {
case x: operand : Print "x"; }
while (top element of stack ≠ "(") do
print topelement and pop;
end while;
pop "(" from stack;
else: while ICP(x) <= ISP(x) do
print top element of stack and pop;
end while;
push x into stack;
end case;
x: get nextchar(E);
end while
~~Example~~: while stack is not emphy do
Print the top element of stack and pop;
end while
End Infix-postfix convertion.

→ Evaluation of Infix to postfix expression:

Here, the postfix expression is evaluated by following criteria that the operand and evaluated results should be stay in stack and the operator should operate the elements (top) and give results.

Algorithm:

Procedure for POSTFIX_EVALUATION:
clear stack;
getnextchar(E) → symb
while symb ≠ $ then {
if symb is operand, print symb to stack
else if symb is operator {
pop the top element of stack as operand 1;
pop the top element of stack as operand 2;
Calculate result = operand 1 operator operand 2;
print result to stack; }
}
getnext char(E) = symb;
end if
end while
end POSTFIX_EVALUATION //.

Example: Let us take ABCD↑*+ A=2 B=-1 C=2, D=3.

OPERATION	STACK	REMARKS
A	A(2)	A is operand
B	A(2) B(-1)	B is operand
C	A(2) B(-1) C(2)	C is operand
D	A(2) B(-1) C(2) D(3)	D is operand
↑	A(2) B(-1) [8]	C&D is popped, evaluated and Pushed
*	A(2) [-8]	[8] and B is poped, evaluated and Pushed
+	[-6]	[-8] and A is poped, evaluated and Pushed
$	x	Result is poped from stack.

4. Define Queue. Write the different types of queue. Explain circular Queue.

Definition of Queue:

Queue is defined as the list of elements arranged in the order, of which the ~~first~~ insertion of element is done at one end called rear and deletion is done at other end called front. It is more applicable to the real life situation. example, a Queue of students.

Types of Queue:

(i) Linear Queue.
(ii) Circular Queue.
(iii) Priority Queue.
(iv) Dequeue are some of types.

(i) Linear Queue:

This type of queue is a linear list of element, where insertion is done at rear and deletion is done at other end called front. It approaches FIFO (First In First Out).

However, the queue gets to call "Queue full", which denotes overflow of elements, sometimes if does not physically full. So it overcomes merit by circular queue.

(ii) Circular Queue:

Circular Queue refers to movements of FRONT & REAR elements which usually be displayed in linear movement, displays in a circular movement over a queue data structure.

(iii) Priority Queue :

Priority Queue is a queue in which the insertion and deletion of elements to any position is based on the property of element called as priority of a task by an element.

Example : A queue of patients is arranged based on priority.

(iv) Dequeue :

Dequeue is a linear set of elements where insertion and deletion of elements in a queue is done at either ends (either front / rear). It uses two variables as LEFT and RIGHT.

CIRCULAR QUEUE :

Circular Queue refers to the queue which FRONT AND REAR variable displayed in a linear movement, displays in a circular movement over a queue data structure.

Premitive Operation of Circular Queue.

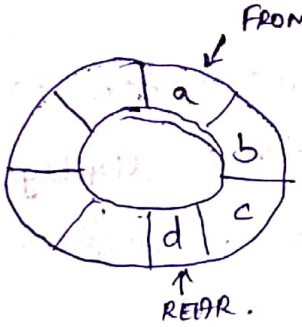
The circular movement of front and rear elements is implemented by mod function which implies circular movement. Let the circular Queue be denote as CIRC_Q[1:n]. The operations are of same as linear queue.

(i) Insertion Algorithm:

```
Procedure ICIRC_Q (Q, FRONT, REAR, n, item)
REAR = (REAR+1) mod n;
if (FRONT = REAR) then
        call QUEUE_FULL;
else Q[REAR] = item;
   ~~FRONT = (FRONT+1) mod n;~~
   ~~item = CIRCQ[item];~~
end ICIRC_Q.
```

(ii) Deletion Algorithm:

```
Procedure DCIRCQ (Q, FRONT, REAR, n)
if (FRONT = REAR) then DCIRC_Q := EMPTY;
FRONT = (FRONT+1) mod n;
item = Q[FRONT];
end DCIRC_Q
```



Example:

OPERATION	QUEUE	value of FRONT	value of REAR	REMARK.
~~Example~~				
Insert a	[a]	(0)	(1)	Insertion successfully
Insert b	[a b]	(0)	(2)	"
Insert c	[a b c]	(0)	(3)	"
Insert d	[a b c]	(0)	(3)	Insertion failure due to overflow of elements.
Delete.	[b c]	(1)	(3)	Deletion Successful.
Delete	[c]	(2)	(3)	"
Insert d	[d c]	(2)	(0)	Insertion Successful.