unit - 3

Stack and queue

D D M M Y Y Y Y

Que- With help of suitable example, Explain Working of PUSH and POP operations of Stack.

⇒

- A stack is a linear data structure that follows the Last-In, First-out (LIFO) Principle, meaning that the last element added to the stack is the first one to be removed.

1] Push operation:-

- The "Push" operation is used to add an element onto the top of the stack
- It increases the stack's size and places the new element on top of the existing element.

- Example:-

Imagine you have an empty stack and you want to push three integers onto it: 10, 20 and 30.

- The stack will look like this after each Push operation.

[30] ← TOP
[20]
[10]

- In this stack, 30 was the last Element added (the top of the stack), and 10 was the first one added (the bottom of the stack).

2] POP operation:-

- The "POP" operation is used to remove the top element from the stack.
- It decreases the stack's size and returns

the element that was Removed.

- Example:-
  if you perform a "pop" operation, it would remove the element 30 from the top of stack, and the stack would look like this.

[20] <— TOP
[10]

- The element 30 has been removed and the top of the stack is now 20.

Que- Explain array based implementation of queues with enque and deque operations.

⇒

- An array-based implementation of a queue is a common way to create a queue data structure.
- In this implementation, a one-dimensional array is used to store the elements of the queue.
- Two pointers, often called "front" and "rear" are used to keep track of the elements positions within the array.

1) Initialization:-
- create an array of a fixed size, which is large enough to accommodate the maximum number of elements you expect in the queue.
- Initialize two pointers: "front" and "rear" to -1 (or 0 if using 0-based indexing).

D D M M Y Y Y Y

② Enqueue (Add an Element):-

- check if the queue is full. You can do this by comparing the "rear" pointer to the size of the array minus one. If they are equal the queue is full.
- If the queue is not full, increment the "rear" pointer by 1 and insert the new element at that position in the array.
- If the "front" pointer is -1, set it to 0.

③ Dequeue (Remove an Element):-

- check if the queue is empty. You can do this by comparing the "front" and "rear" pointers. If "front" is greater than "rear" the queue is empty.
- If the queue is not empty, remove the element at the "front" position and increment the "front" pointer by 1.
- If after dequeuing an element, "front" becomes greater than "rear", it means the queue is now empty. So you can reset both "front" and "rear" to -1.

Que- Distinguish between stack and queue.
→

STACK	QUEUE
1] It is LIFO (Last-In-First-out) data structure	1] It is FIFO (First-In-First out) data structure.
2] Insertion and deletion take place at only one end called top.	2] Insertion takes place at rear and deletion takes place at front.
3] It has only one Pointer variable	3] It has two pointer variables.
4] No memory wastage	4] Memory wastage in linear queue.
5] operations:- 1] Push() 2] POP()	5] operations:- 1] enqueue() 2] dequeue()
6] In computer system it used in procedure calls	6] In computer system it is used time/resource sharing.
7] Plate counter at marriage reception is an example of stack	7] Student standing in a line at fee counter is an example of queue.

Que- Explain stack and operations Algorithm.

⇒

Stack :-

- A stack is a Abstract data type.
- A stack is a data structure that follows the Last-In-First-out (LIFO) Principle, meaning that the last element added to the stack is first one to be removed.

Stack operations :-

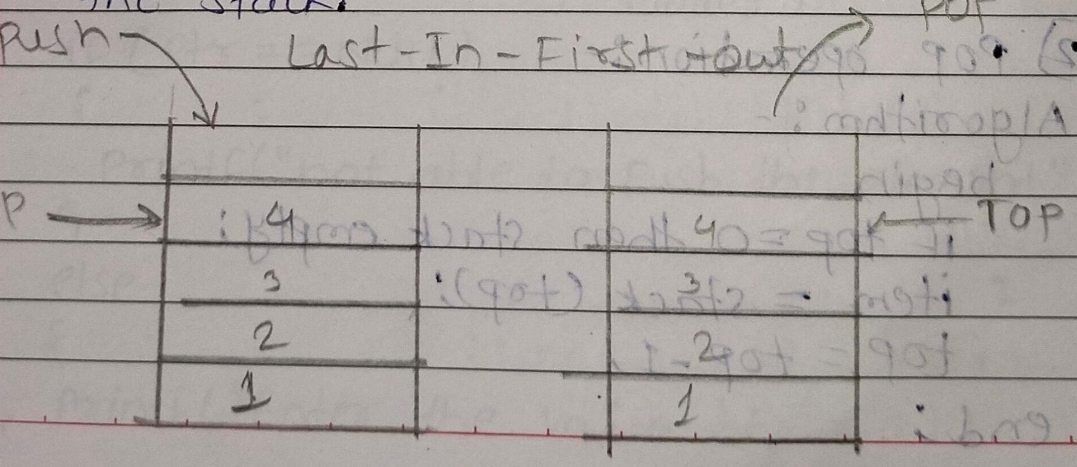
1) Push : This operation is used to add an element to the top of the stack.

2) Pop :- This operation removes and returns the element at the top of the stack.

3) Peek (or TOP) :- This operation allows you to view the element at the top of the stack without removing it.

4) isEmpty :- checks if the stack is empty.

5) Size :- Returns the number of elements in the stack.



1] Array implementation of stack:-

a) Push operation:-

Algorithm:-

```
begin
  if top = n then stack full
  top = top + 1
  stack (top) = item;
end
```

implementation of Push algorithm in c language

```
void Push (int val, int n) // n is size of stack
{
  if (top == n)
  printf ("\n overflow");
  else
  {
  top = top + 1;
  stack [top] = val;
  }
}
```

2) Pop operation:-

Algorithm:-

```
begin
  if top = 0 then stack empty;
  item = stack (top);
  top = top - 1;
end;
```

Implementation of Pop algorithm using C language.

```
int Pop()
{
 if(top==-1):
 {
  printf("underflow");
  return 0;
 }
 else
 {
  return stack[top--];
 }
}
```

2) Linked List implementation of stack:-

a) Push operation:-

```
Void Push()
{
  int val;
 struct node *ptr=(struct node*)malloc(sizeof(struct
                  node));
 if(ptr==NULL)
 {
  printf("not able to push the element");
 }
 else
 {
 printf("Enter the value");
```

```
scanf("%d", &val);
if(head == NULL)
{
  ptr -> val = val;
  ptr -> next = NULL;
  head = ptr;
}
else
{
  ptr -> val = val;
  ptr -> next = head;
  head = ptr;
}
printf("Item Pushed");
}
```

② Pop :-

```
Void pop()
{
  int item;
  struct node *ptr;
  if(head == NULL)
  {
    printf("underflow");
  }
  else
  {
  item = head -> val;
  ptr = head;
```

```
        head = head -> next;
        free(ptr);
        printf("Item popped");
    }
}
```

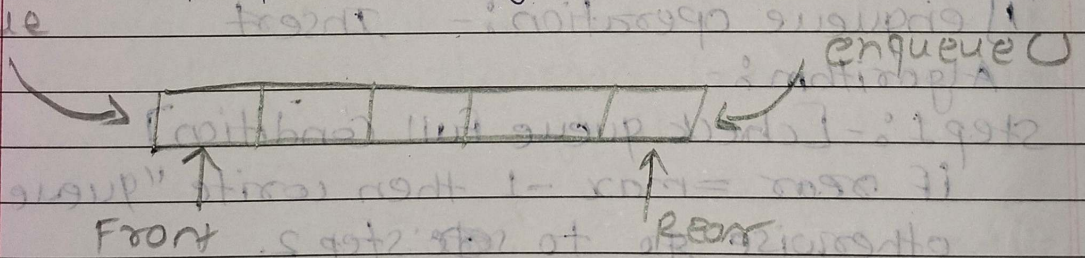
Que- Explain queue and its operations algorithm.

⇒

Queue :-

- ordered collection of homogeneous elements
- Non-Primitive linear data structure.
- A new element is added at one end called rear end and the existing elements are deleted from the other end called front end.
- This mechanism is called First-In-First-out. (FIFO).

e.g :- people standing in queue for Movie Ticket.



- enqueue() is the operation for adding an element into queue.
- dequeue() is the operation for removing an element from queue.
- queue is abstract data type.

• Elements of queue :-

1] Front :-
- This end is used for deleting an element from a queue.
- Initially front end is set to -1.
- Front end is incremented by one when a new element has to be deleted from queue.

2] Rear :-
- This end is used for inserting an element in a queue.
- Initially rear end is set to -1.
- rear end is incremented by to be inserted in queue.

• operations :-

1] enqueue operation :- Insert
Algorithm :-
step 1 :- [check queue Full condition]
if rear = max -1 then write "queue is Full" otherwise go to step 2.
step 2 :- [increment rear point]
rear = rear + 1;
step 3 :- [insert element]
q[rear] = Data
step 4 :- [check front pointer]
if front = -1 then assign front = 0
step 5 :- End.

② dequeue operation:-

Algorithm:-

step 1:- [check queue empty condition]
if front = -1 then write "queue is empty"
otherwise go to step 2.

step 2:- [copy data]
data = q[front]

step 3:- [check front and rear pointer]
if front = rear then
front = rear = -1
otherwise
front = front + 1

step 4:- End.

Que- Give Applications of Queue and Stack.

⇒

Application of Stack :-

1] Function call stack :-
- Stacks are used in programming languages to manage function calls.
- Each function call is pushed onto the stack, and when a function returns, its popped from the stack.

2] Undo mechanisms :-
- stack are employed in software application to implement undo functionality, allowing users to revert to previous states.

3] Expression Evaluation :-
- Stacks are used to evaluate expressions, particularly in programming languages where operators and operands are pushed and popped accordingly.

4] Backtracking :-
- Stacks are used in algorithms like depth-first search (DFS) for backtracking through a problem space.

5] Memory management :-
- Stacks are used in managing memory allocation and deallocation, particularly in systems programming.

Applications of queue:-

1] Job Scheduling:-
- Queues are used in operating systems to schedule tasks and manage system resources in a fair and orderly manner.

2] Print queue:-
- Queues are used to manage Print jobs in a printer's queue, ensuring that Print requests are processed in the order they are received.

3] Breadth-First search (BFS):-
- Queues are used in graph algorithms like BFS to explore nodes level by level, which is particularly useful in finding the shortest path in unweighted graphs.

4] Task management:-
- In concurrent programming, queues are used to manage and distribute tasks among multiple threads or processes.

5] Data Buffering:-
- Queues are employed in scenarios where data needs to be buffered and processed asynchronously, such as in message queuing system.

Types of ~~queue~~ queue :-

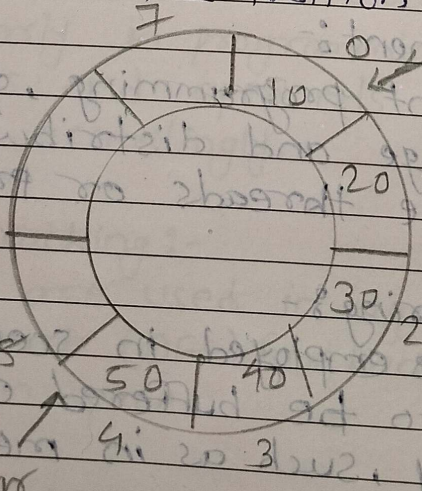
1) circular queue
2) pequeue (Double Ended queue)
3) Priority queue.

Que- Explain circular queue with ~~aim~~ Algorithms.

⇒

Circular queue :-

- A queue, in which the last node is connected back to the first node to form a cycle is called as circular queue.
- circular queue are the queues implemented in circular form rather than in a straight line.
- circular queues overcome the problem of unutilized space in linear queue implemented as an array.
- The main disadvantage of Linear queue using array is that when elements are deleted from the queue, new elements cannot be added in their place in the queue i.e. the position cannot be reused.



- Circular queue Implementation:-
  - After rear reaches the last position i.e. MAX-1 in order to reuse the vacant positions, we can bring rear back to the 0th position, if it is empty, and continue incrementing rear in same manner as earlier.
  - Thus rear will have to be incremented circularly.
  - for deletion, front will also have to be incremented circularly.

- Enqueue (Insert) operation on circular queue:

Step 1:- check for queue full
If rear = max-1 and front = 0 or if front = rear+1 then circular queue is full and insertion operation is not possible otherwise go to step 2.

Step 2:- check position of rear pointer
If rear = max - 1
then set rear = 0 otherwise increment rear by 1.
rear = (rear + 1) % MAX

Step 3:- Insert element at the Position Pointed by rear pointer
q[rear] = Data

Step 4:-
check the position of front pointer
If front = 1 then set front as 0.

- Dequeue (Delete) operation on circular queue:-

Step 1:- check for queue empty if(Front = -1) then circular queue is empty and deletion operation is not possible.
otherwise go to step 2.

step 2:- check for position of front and rear pointers.
if Front = rear then
Data = q[front];
set front = rear = -1

step 3:- check position of front
if front = max - 1
Data = q[front];
then set front = 0;
otherwise
Data = q[front];
Front = (front + 1) % MAX

Explain priority queue with ~~operations~~ example ~~Algorithm~~.

⇒

Priority Queue:-

- A Priority queue is a collection of elements where each element is assigned a priority and the order in which elements are deleted and processed is determined from the following rules:-

1] An element of higher priority is processed before any element of lower priority.

2] Two elements with the same priority are processed according to the order in which they are added to the queue.

- In priority queue node is divided into three parts.

Store Data ← | INFO | PRIORITY | NEXT | → Point to next node

- An example where priority queue are used is in operating systems.
- The operating system has to handle a large number of jobs
- These jobs have to be properly scheduled.
- The operating system assigns priorities to each type of job.
- The jobs are placed in a queue and the job with the highest priority will be executed first.

- Application of Priority queue:-

1] It is used in the Dijkstra's shortest Path algorithm

2] It is used in Prim's algorithm

3] It is used in heap sort

4] It is also used in operating system like Priority scheduling, Load balancing and interrupt handling.