Q1. Define Stack as an ADT.

Ans A Stack is a type of data structure that works on the principle of LIFO (Last in, First out). This means the last item added to the stack is the First one to be removed. Imagine a stack of plates: you can only add a new plate on top and remove the top plate First.

- Push: Add an item to the top the stack.
- Pop: Remove the item from the top of the Stack.
- Peek/Top: See the top item without removing it.

Q2. Write the algorithms to PUSH and POP element Stack.

Ans PUSH

```
Void push() {
        int x;
        Printf("Enter data:");
        ScanF("%d", &x);
        if (top == n-1) {
        Printf("Stack overflow\n"); }
        else { top++;
            Stack[top] = x;
        Printf("%d Pushed to Stack\n", x);
        } }
```

notessociety1@gmail.com

POP

```
Void pop() {
        if (top == -1) {
            printf("Stack underflow\n");
        } else { int popped_value = stack[top];
            top--;
            printf("%d popped From Stack \n", popped_value);
        }
    }
```

Q.3. Write any four Applications of Stack.

Ans i) Expression Evaluation and Conversion : Stacks are used to evaluate Mathematical expressions (like postfix) and convert expressions between infix, postfix, and prefix notations.

ii) Undo and Redo operations in Software : In text editors and software, stacks are used to store changes, allowing you to undo or redo your last actions.

iii) Function Call Management in programming : Stacks keep track of function calls in programs. When a function call another function, the current function's details are pushed onto the stack.

iv) Balancing Parenthese and Syntax parsing : Stacks help check if parenthese { }, [ ], ( ) in code are balanced and assist in parsing Programming languages.

PIMPRI CHINCHWAD UNIVERSITY

Q4. What are Stack empty and Stack full Conditions?

Ans

Empty

A Stack is empty when there are no item in it. This happens when the top pointer/index is set to -1 (or a similar value that represents no elements).

→ top == -1

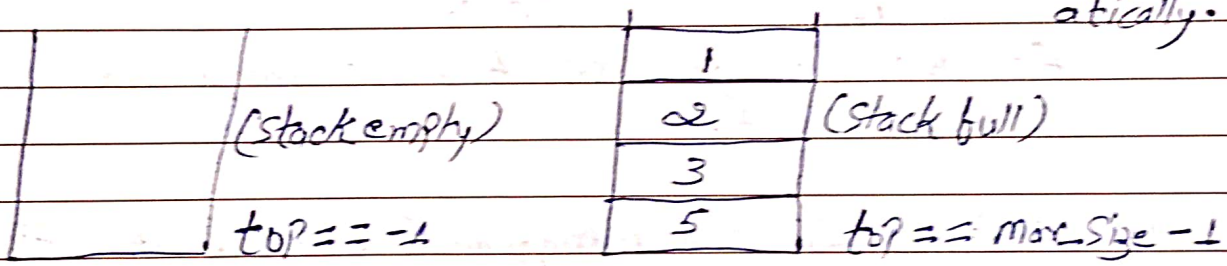
Full

A Stack is full when there is no more space to add a new item. This happens when the top pointer/index reaches the maximum size of the Stack.

→ top == max_size - 1

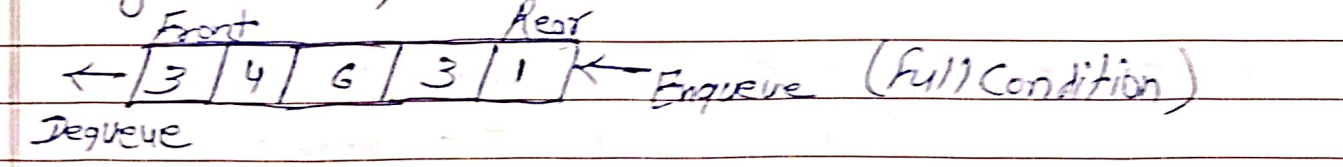
Q5. What are Stack empty and Stack full Conditions? Show diagrammatically.

Ans

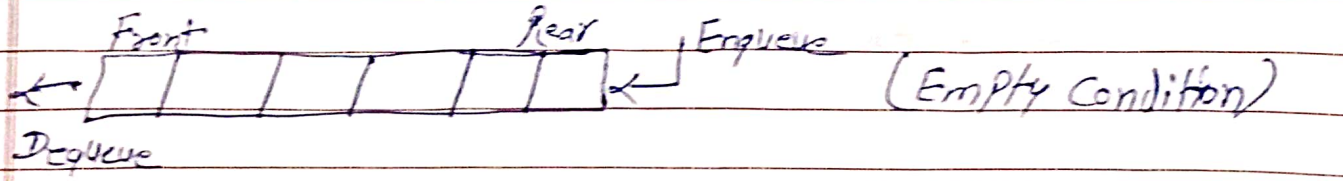


Q6. What are queue empty and queue full Conditions? Show diagrammatically.

Ans



→ Rear == Max_Size - 1



→ Front == -1 && Rear == -1

Q7. List the type of queue.

Ans i) Simple Queue (Linear Queue): Element inserted at the rear and remove from front. (FIFO)

ii) Circular Queue: Connects the last position back to the First position.

iii) Priority Queue: Each element is assigned a priority.

iv) Double-Ended Queue (Deque): Elements can added or removed from both ends (front and rear).

Q8. Write any four applications of queue.

Ans i) Task scheduling: Operating Systems to schedule tasks like processes or print jobs, Task are handled in the order they are received (FIFO)
Ex: A printer processes documents one by one in the order they are sent.

ii) Customer Service Management: In banks, ticket counters, or call centers, queue are used to manage customer service requests in the order they arrive.
Ex: A ticket counter where people are served one by one.

iii) Data Buffering: Queue are used in buffering, such as Streming audio or video. Data packets are queued and processed in sequence to ensure smooth playback.
Ex: Watching a YouTube video that loads chuncks of data in a queue.

iv) Simulation and Modeling: Queue are used in simulations to model real-life scenarios like traffic systems, airport runway, or queues in supermarkets.
Ex: Simulating how cars wait at a traffic signal.

8 g. Write the algorithms to INSERT and DELETE elementin Queue.

Ans

```
Void enqueue() {
        int x;
        Printf("Enter data:");
        Scanf("%d", &x);
        if(rear == n-1) {
            Printf("Queue overflow\n"); }
        else { if(front == -1) {
                front = 0; }
            rear++;
            queue[rear] = x;
            Printf("%d added to queue\n", x);
        } }
Void dequeue() { if(front == -1 || front > rear) {
            Printf("Queue underflow\n");
        } else {
            int dequeued_value = queue[front];
            front++;
            Printf("%d removed From queue\n", dequeue
                -value)
            if(front > rear) {
                front = rear = -1; }
        } }
```

Q10. Convert the following infix to postfix expression.
A+(B*C+D)/E

Ans

Infix	Stack	Postfix
A		A
+	+	A
(	+(	A
B	+(	AB
*	+(*	AB
C	+(+	ABC*
+	+(+	ABC*
D	+(+	ABC*D
)	+	ABC*D+
/	+/	ABC*D+
E	+/	ABC*D+E

postfix : ABC*D+E/+

Q11. Convert following infix to postfix and postfix to infix.
expression A+B-C*D/E+F

Ans

Infix	Stack	Postfix
A		A
+	+	A
B	+	AB
-	-	AB+
C	-	AB+C
*	-*	AB+C
D	-*	AB+CD
/	-/	AB+CD*
E	-/	AB+CD*E

PCET's Pimpri Chinchwad University
Learn | Grow | Achieve

+	- +	AB+CD*E/
F	- +	AB+CD*E/F+-

Postfix expression : AB+CD*E/F+-

⇒ Infix to Prefix

Postfix	Stack	Prefix
F		F
+	+	F
E	+	FE
/	+/	FE
D	+/	FED
*	+*	FED/
C	+*	FED/C
-	+-	FED/C*
B	+-	FED/C*B
+	++	FED/C*B-
A		FED/C*B-A++

Prefix: ++A-B*C/DEF

8 12. Evaluate the following Postfix Expression 823*+7/1-

Ans

8	8
23	23
*	8*23 = 18
+	184+0 = 19
7	184, 7
/	1847 = 26.2857
1	26.2857, 1
-	26.2857-1 = 25.2857

PIMPRI CHINCHWAD UNIVERSITY

PCET's Pimpri Chinchwad University
Learn | Grow | Achieve

∴ Final = 25.2857

Q13. Evaluate the following Prefix Expression. * + 38 - 21

* + 38 - 21	1	1
* + 38 - 21	2	[1, 2]
* + 38 - 21	-	1
* + 38 - 21	8	[1, 8]
* + 38 - 21	3	[1, 8, 3]
* + 38 - 21	+	[1, 11]
* + 38 - 21	*	[11]

∴ final ans => * + 38 - 21 = 11.

PIMPRI CHINCHWAD UNIVER