• Runtime Complexity for heap sort, merge sort - O (n log n).

Que 6 Write the procedure to perform push and pop opration on Stack.

→

Push an element

~~Start~~ • Step 1 - START

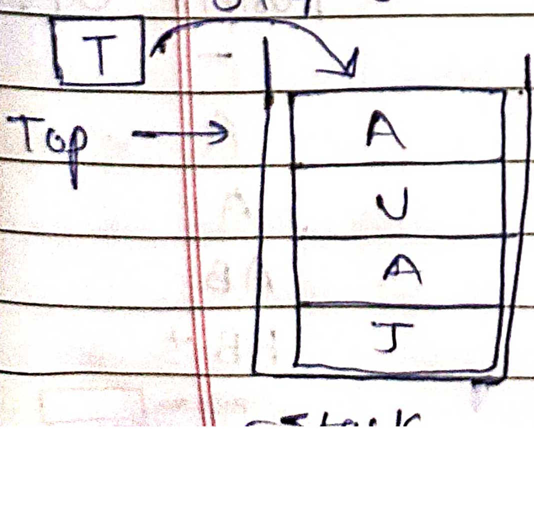
• Step 2 - Store the element to push into array.

• Step 3 - check if top == (max size -1) then stack is full else go to step 4

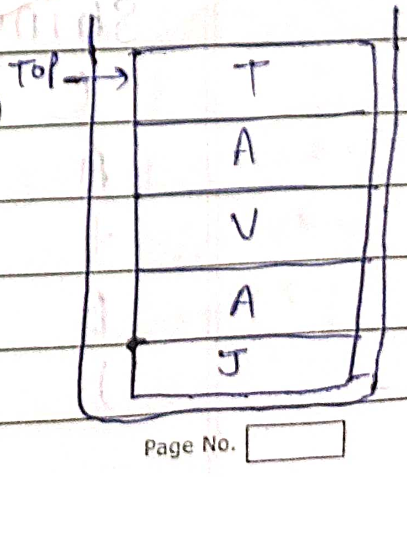
• Step 4 - increment top as top = top + 1

• Step 5 - Add element to the position stack[top] = num.

Step 6 - STOP



Push opration



Date ___ / ___ / ______

Pop opration can be peryormed in the below steps.

Step 1 - Checks stack has some element or stack has empty

Step 2 - If the stack has no element means it is empty then display " underflow "

Step 3 - If the stack has element some element accessed the data element at which top is pointing.

Step 4 - Decreses the value of top by 1

Step 5 - Pop opration peryormed.

ue 8 Write a note on i) priority queue
ii) Multiple ~~queue~~ . Stacks.

→

i) Priority Queue :-

A priority queue is a type of queue that arranges element based on their values, Elements with higher priority values are typically removed before elements with lower priority values.

In a priority queue, each element has a priority value associate with it. When you add an element to the queue it is inserted in position based on its priority value for exmple if you add an element with high priority value to a priority queue it may be inserted near the front of queue

Page No.

while an element with low priority value may be inserted neare the back.

ii) Multiple Stacks :-

A single stack is sometime not sufficient to store a large amount of data to overcome this problem we can use multiple stacks for this we have used a single array having more than one stack. The array is divided for multiple stacks.

Suppose there is an array STACK[n] divided into two stack STACK A and STACK B, where n = 10.

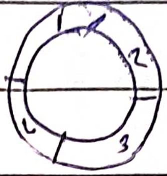
• Stack A expands from the left to Right i.e from ~~others~~ 0th element.

• ~~SAT~~ STACK B expand from the right to the left i.e from 10th element

• ~~STACK B~~

• The combined size of both STACK A and STACK B never exceeds 10

Que 10 Explain and illustrate the Concept of circular queue.



→

There was ~~one limitation~~ one limitation in the array implementation of queue. If the rear reaches to the end position of the queue then there might be possiblity that some vaccant spaces are left in the beginning which cannot be utilized so to overcome such limitation the concept of circular queue was introduced. The rear is at last position of queue and front is pointing somewhere rather than the 0th position. There are only two element and other three position are empty The rear, is at last position of the Queue; if we try to insert element then it will Show that there are no empty spaces in the queue. There is one sol^n to avoid such wastage of memory space by shifting both the elements at left and adjust front and rear end accordingly

[illegible]

Page No.

Date ___/___/___

Que 9 Explain polish Notation Convert the following notation to postfix using stack. show all stack position.

A+(B*C-(D/E↑F)*G)*H

→

Polish Notation also known as normal polish Notation, Lukasiewicz notation, warsaw notation, polish prefix notation is a mathematical notation in which oprators are placed between operands as well as reverse polish notation (RP

A+(B*C-(D/E↑F)*G)*H

String	Stack	Postfix Exp
A		A
+	+	A
(	+(	A
B	+(	AB
*	+(*	AB
C	+(*	ABC
−	+(−	ABC*
(	+(−(	ABC*
D	+(−(	ABC*D
/	+(−(/	ABC*D
E	+(−(/	ABC*DE
↑	+(−(↑	ABC*DE/
F	+(−(↑	ABC*DE/F

Date ___/___/___

)	+(−	ABC*DE/f↑
*	+(−*	ABC*DE/f↑
G	+(−*	ABC*DE/F↑G
)	+	ABC*DE/f↑G*−
~~*~~ *	+*	ABC*DE/f↑G*−
H	+*	ABC*DE/f↑G*H
		ABC*DE/f↑G*H*+

A Doubly Ended Queue (also known as Deque) is a linear data structure that allows insertion and deletion of elements from both ends. It can be visualized as a combination of a queue and a stack. In a doubly ended queue, elements can be inserted or deleted from either the front or the rear end.

3. Pop Operation:
    - Check if the stack is empty. If it is empty, display an underflow message and terminate the operation.
    - If the stack is not empty, retrieve the value or element at the top of the stack.
    - Remove the value or element from the top of the stack.
    - Update the top pointer or index to reflect the new top of the stack.
4. Repeat steps 2 and 3 as needed for additional push and pop operations.

1. Initialize an empty stack. This can be represented as an array, a linked list, or any other suitable data structure.
2. Push Operation:
    - Check if the stack is full (if there is a maximum capacity). If it is full, display an overflow message and terminate the operation.
    - If the stack is not full, prompt the user to enter the value or element to be pushed onto the stack.
    - Add the value or element to the top of the stack.
    - Update the top pointer or index to reflect the new top of the stack.