EXPERIMENT NO - 01

* Write a program to implement stack using arrays.

* Theory -

i) A stack is an abstract data type, that is popularly used in most programming language.
ii) It is named stack because it is similar operation as the real world stack, for ex. a pack of cards.
iii) The stack follows the LIFO structure where the last element inserted would be the first element deleted.
iv)

Stack representation -

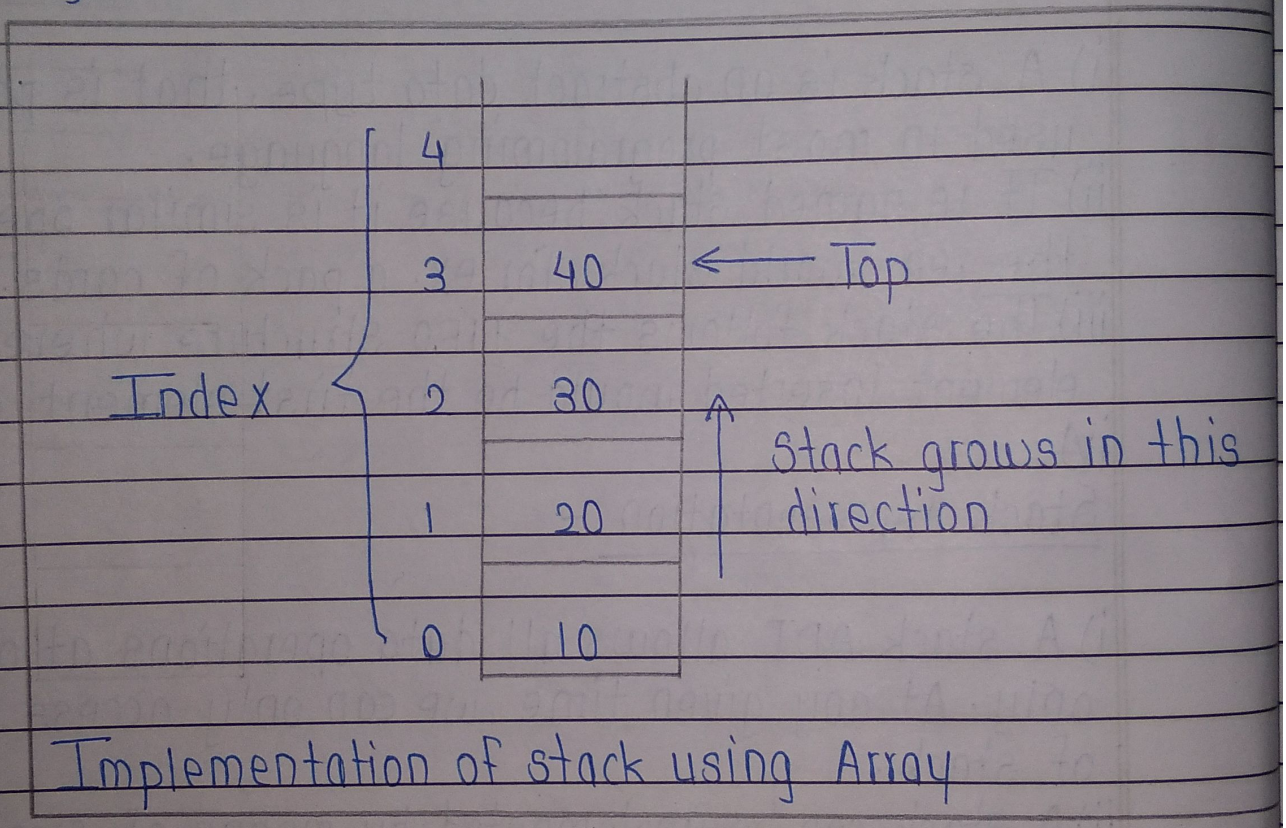
i) A stack ADT allows all data operations at one end only. At any given time, we can only access top element of stack.
ii) A stack can be implemented by means of Array, Structure pointer and linked list.
iii) Stack can be either be a fixed size one or it may have a sense of dynamic resizing.
iv) Here we are going to implement stack using arrays, which makes it a fixed size stack implementation.

Basic operation on stack -

i) push ( ) - For adding the element in stack
ii) pop ( ) - For deleting the element from stack

iii) peek ( ) - Return the element of a given positio
iv) isFull ( ) - Check whether the stack is full or not
v) isEmpty ( ) - Check whether the stack is empty or

Diagram -



Implementation of stack using Array

1] Push - When we insert an element in stack then t operation is known as push. If the stack is full then overflow condition occurs.

2] Pop - When we delete an element from stack th the operation is known as pop. If the stack is empty then underflow condition occurs.

NUTAN COLLEGE OF ENGINEERING AND RESEARCH
योगः कर्मसु कौशलम्
Skilled Engineers for Industry

Conclusion -

i) Implementing a stack using arrays is a straight forward and efficient approach.
ii) Arrays provide constant time complexity for push and pop operations.
iii) Memory usage is optimized as array have lower overhead compared to linked list.
iv) Accessing elements by index in an array allows for easy manipulation of stack.

NUTAN COLLEGE OF ENGINEERING AND RESEARCH
योगः कर्मसु कौशलम्
Skilled Engineers for Industry

EXPERIMENT NO 02

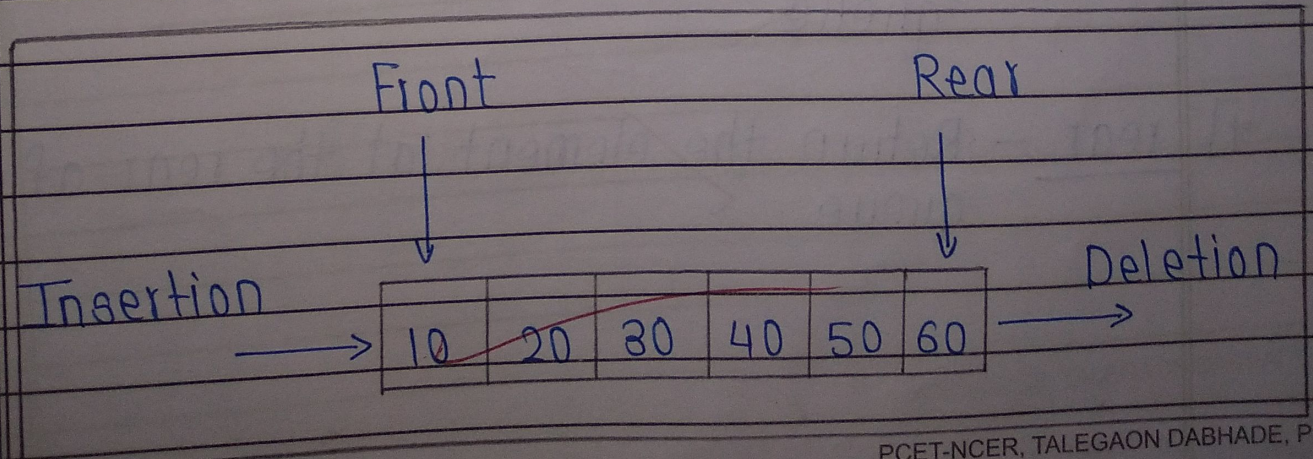
* Write a program to implement queue using arrays.

* Theory -

i) Queue, like stack, is also an abstract data type.
ii) The thing that make queue different from stack is that a queue is open at both it ends.
iii) It follows FIFO structures i.e. data item inserted first will also be accessed first.
iv) The data is inserted into the queue through one end and deleted from it using the other end.
v) A real world example of queue can be a single-lane one way road, where the vehicle enters first, exits first.

Representation of queue -

i) A queue ADT implemented by using, array, linked list or pointers.
ii) Here we are going to implement a queue using a one-dimensional array.



Basic operation -

i) enqueue ( )
ii) dequeue ( )
iii) peek ( )
iv) isFull ( )
v) isEmpty ( )

1] Enqueue - Adds the elements to the end of the queue.

2] dequeue - Remove the elements from front of the queue.

3] peek - Return the value of given position

4] isEmpty - Check whether the queue is empty or not.

5] isFull - Check whether the queue is full or not

6] front - Return the element at the front of the queue

7] rear - Return the element at the rear of the queue.

NUTAN COLLEGE OF ENGINEERING AND RESEARCH
Skilled Engineers for Industry

Conclusion -

i) Implementing a queue using an array offers simplicity and efficient memory usage.
ii) Arrays provide a fixed capacity, which can limit the number of element in the queue.
iii) Enqueue and dequeue operations can be efficiently performed by shifting element within the array.

EXPERIMENT NO-03

* Implementation of stack using linked list.

* Theory -

i) To implement a stack using singly linked list concept all the singly linked list operation should be performe based on stack operation LIFO.
ii) With the help of that knowledge, we are going to implement a stack using linked list.

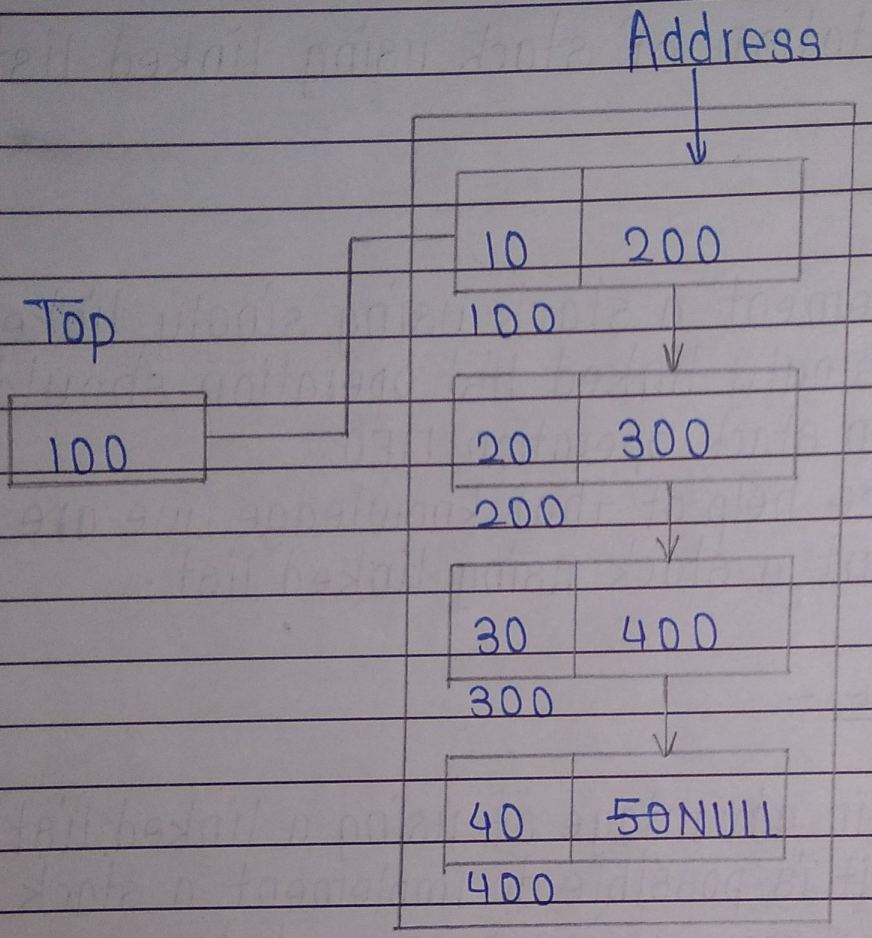
Advantage -

i) The main advantage of using a linked list over arrays is that it is possible to implement a stack that can shrink or grow as much as needed.
ii) Using a array will put a restriction on the maximum capacity of the array which can lead to stack overflow.
iii) Here each new node will be dynamically allocated. So overflow is not possible.

Operation on stack -

i) push ( )
ii) pop ( )
iii) isEmpty ( )
iv) display ( )
v) peek ( )

Diagram -



Implementation of stack using linked list

Main operation on stack -

1] Push operation -

i) Create a node and allocate memory.
ii) If the list is empty, then item is pushed as start node of list. This include assigning value to data part and null address part.
iii) If there are node in list, then we have to add

new element in beginning of list.
iv) For that assign address the starting element to address field of new node and make new node the starting node.

2] Pop operation -

i) Check the underflow condition.
ii) Adjust the head pointer accordingly. Element are popped only from one end.
iii) Therefore the value stored in head pointer must be deleted and the node must be freed.
iv) The next node of the head node now become head node.

Conclusion -

i) Using a linked list for a stack offers flexibility and efficient memory allocation.
ii) Linked list allows for easy insertion and deletion of elements at the top of the stack.
iii) It can accomodate an arbitary number of elements, providing scalability.

NUTAN COLLEGE OF ENGINEERING AND RESEARCH
Skilled Engineers for Industry

EXPERIMENT NO - 04

* Write a program to implement queue using linked list.

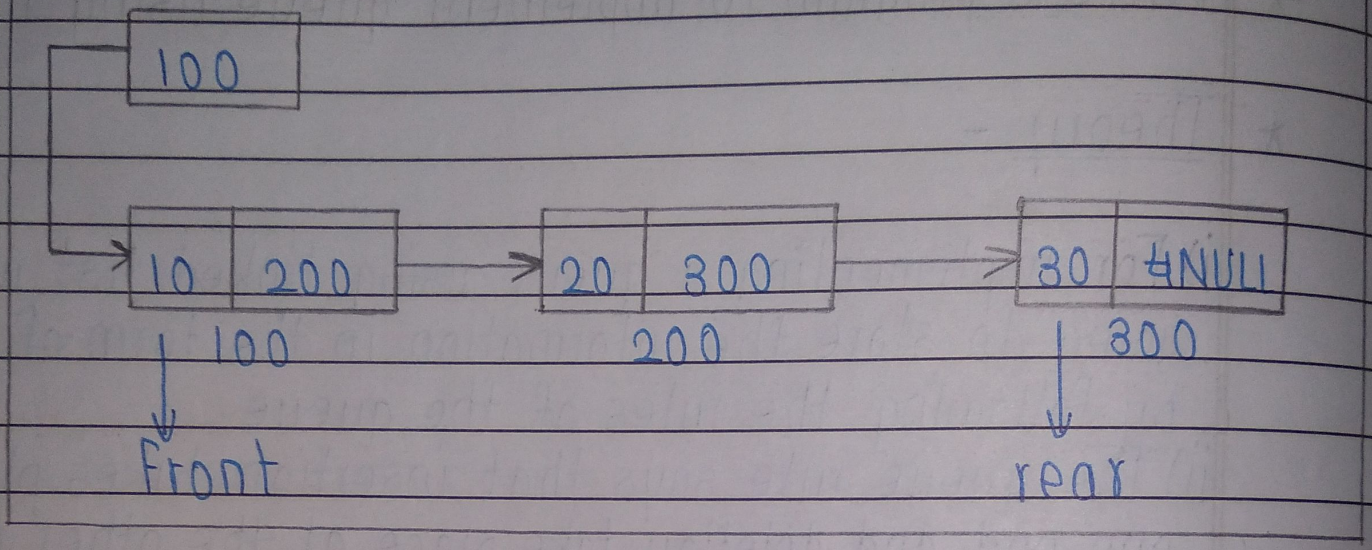
* Theory -

i) For implementing a queue using linked list we are going to store the information in the form of nodes by following the rules of the queue.
ii) The queue rule says that insertion takes place at one end and deletion take place at the other end.
iii) The benefit of implementing a queue using a linked list over array is that it allows to grow the queue as per the requirement, i.e. memory can be allocated dynamically.
iv) A linked queue consist of two pointers, i.e., the front pointer and the rear pointer.
v) Insertion is performed at rear end, whereas deletion is performed at the front end of the queue.
vi) If both front and rear point to the NULL, it signifies that the queue is empty.

Operation on the linked queue -

i) Insertion
ii) Deletion
iii) isEmpty ()
iv) display ()

Diagram –



Main operations on queue –

1] Enqueue –

i) Declare a newnode and allocate memory.
ii) If front == NULL, make both front and rear point to newnode. Otherwise add the new node in rear→next make the newnode as the rear node.

2] Dequeue –

i) Check whether the queue is empty or not.
ii) If it is empty queue, we can't dequeue the element
iii) Otherwise, make the front node points to the next node i.e. front = front→next.

NUTAN COLLEGE OF ENGINEERING AND RESEARCH
योगः कर्मसु कौशलम्
Skilled Engineers for Industry

Conclusion -

i) Implementing a queue using a linked list provides flexibility in terms of dynamic memory allocation.
ii) Linked list can adjust their size easily, allowing for efficient enqueue and dequeue operation.
iii) Memory usage can be optimized as linked list only require memory for the elements and pointers.
iv) Insertion and deletion operations are efficient even when the queue is large.

EXPERIMENT NO - 05

* Write a program to convert a infix expression to postfix form using stacks.

* Theory -

When the operator is written in between the operands, then it is known as infix notation.

Example - (p+a)*(r+s)

Syntax - <operand><operator><operand>

Postfix expression -

The postfix expression is an expression in which the operator is written after the operand.

Example - ab+

Key points -

i) Operations are performed in order in which they have written from left to right.
ii) Do not require paranthesis.
iii) We do not need to apply operator precedence rules and associativity rules.

Algorithm -

i) Scan the expression from left to right until we encounter any operator.
ii) Perform the operation
iii) Replace the expression with its compound valu
iv) Repeate the steps from 1 to 3 until no more operand exist.

Conclusion -

i) In conclusion the stack data structure is the be method for converting an infix expression to a po expression.
ii) The algorithm for converting infix to postfix nota involves using a stack to keep track of operators their precedence.
iii) Also the time complexity of infix to postfix expression conversion algorithm is mathematically found to be O(n).
iv) Thus using the stack data structure is the best method for converting it.