Exercise 2 Stack

2. Write a C++ program for implementation of stack Using
a) Array b) Linked List

Objective: The objective of this exercise is to enable you to program a stack and to perform operation on it.

Procedure and description:
A stack is a data structures in which insertion and deletion of items are made at the one end, called the top of the stack. We have two basic operations in stack they are push and pop

Push: Used to add an item into stack

Pop: Used to delete an item from stack

Algorithm:
a) Implementing of stacks using array:
1. Push Operation:
The push operation is used to add an item into a stack. Before executing push operation one must check for the OVERFLOW condition, i.e. check whether there is room for the new item in the stack.

Notations Used:

STACK	:	An Array containing the elements of the Stack
TOP	:	Its stack pointer variable (which contains current (top) location of stack)
ITEM	:	New data to be added to the list
MAXSTK	:	End of stack pointer (maximum number of elements that can be held by stack)

Algorithm:
PUSH (STACK, TOP, MAXSTK, ITEM)
Step 1: [Stack already full?]
If TOP = MAXSTK, then: print: OVERFLOW, and Return.
Step 2: Set TOP: = TOP + 1. [Increases TOP by 1]
Step 3: Set STACK [TOP]:= ITEM. [Insert ITEM in new TOP position]
Return.

As per the procedure first it checks for the OVERFLOW condition. Since the stack is not full the TOP pointer is incremented by 1, TOP= TOP + 1. So,

now TOP points to a new location and then the ITEM is inserted into that position.

2. Pop Operation:

The pop operation is used to remove an item from a stack. Before executing the pop operation one must check for the UNDERFLOW condition, i.e. check whether stack has an item to be removed.

Algorithm:

POP (STACK, TOP, ITEM)
Step 1: [Stack is empty?]
If TOP = 0, then: Print: UNDERFLOW, and Return.
Step 2: Set ITEM: = STACK [TOP]. [Assign Top element to ITEM]
Step 3: Set TOP: = TOP – 1. [Decreases Top by 1]

Return

As per the procedure, first it checks for the underflow condition. Since stack is not empty the top element in the stack is assigned to ITEM.

Expected Output:

After executing program. Enter INPUT, Enter size of array and Enter choice either insertion or deletion operation. For insertion enter integer number into stack. For deletion just enter choice as delete. For better understanding see below pictorial representation.

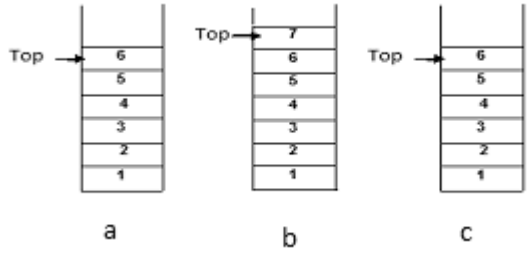

Fig: a) Initial Stack with elements b) After inserting element 7 at top
c) After deletion element 7 from top

b) Stack using Linked list

Procedure and description:

Linked list implementation of stack uses singly linked list or one- way list where the DATA field contains the ITEM to be stored in stack and the link field contains the pointer to the next element in the stack. Here TOP points

to the first node of linked list which contains the last entered element and null pointer of the last node indicates the bottom of stack

Algorithm: The steps for this exercise are given below:

Node : linear collection of data elements. Each node divides into two parts DATA and LINK
DATA : First part called DATA contains data value
LINK : Second part called LINK, contains address
TOP : Its stack pointer variable (which contains location of stack)
START : The address of the first node
AVAIL : Its link pointer variable
ITEM : New data to be added to the list

Definition of node:

```
Class Node
Start
    INT     ITEM
    Node   *LINK
End
```

1. Push operation

Push operation is performed by inserting a node into the start of the list

Algorithm:

```
PUSH (DATA, LINK, TOP, AVAIL, ITEM)
Step 1:  [Available space?]
If AVAIL = NULL, then Write OVERFLOW and Exit
Step 2:  [Remove first node from AVAIL list]
Set NEW: = AVAIL and AVAIL: = LINK [AVAIL].
Step 3: Set DATA [NEW]:= ITEM [Copies ITEM into new node.]
Step 4: Set LINK [NEW]:= TOP [New node points to
                the original top node in the stack]
Step 5: Set TOP = NEW [Reset TOP to point to the new node
            at the    top of the stack]
Exit.
```

2. Pop Operation

Pop operation is performed by removing the first or the start node of a linked list.

Algorithm:

POP (DATA, LINK, TOP, AVAIL, ITEM)

Step 1: [Stack is empty?]
If TOP = NULL then Write: UNDERFLOW and Exit.

Step 2: Set ITEM: = DATA [TOP] [copies the top element of stack into ITEM]

Step 3: Set TEMP: = TOP and TOP = LINK [TOP]
[Remember the old value of the TOP pointer in TEMP and reset TOP to point to the next element in the stack.]

Step 4: [Return deleted node to the AVAL list]

Step 5: Set LINK [TEMP] = AVAIL and AVAIL = TEMP.

Exit.

Expected Output:

Insertion and deletion can be performed at one end only (top end).

After executing program. Enter Input choice either insertion or deletion.

For better understanding see below pictorial representation.

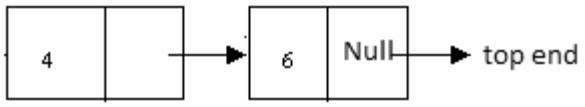

Fig.: Stack Initial linked list

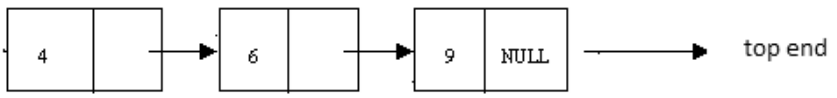

Fig.: After inserting element 9 at top end

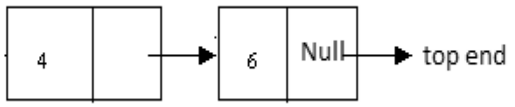

Fig.: After deleting element 9 at top end