**Experiment No 7:**

**A program for Insertion Sort**

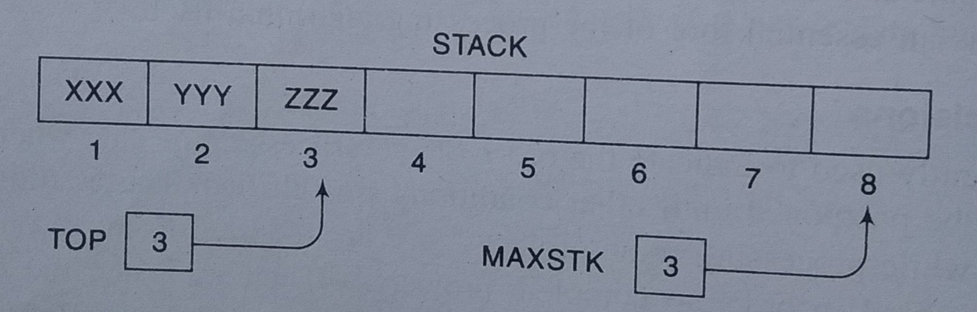
**Aim:** A program for pushing and popping an element into a STACK.

**Theory:**

Stack is an important data structure which stores its elements in an ordered manner. Take an analogy of a pile of plates where one plate is placed on top of the other. A plate can be removed from the topmost position. Hence, you can add and remove the plate only at/from one position that is, the topmost position.

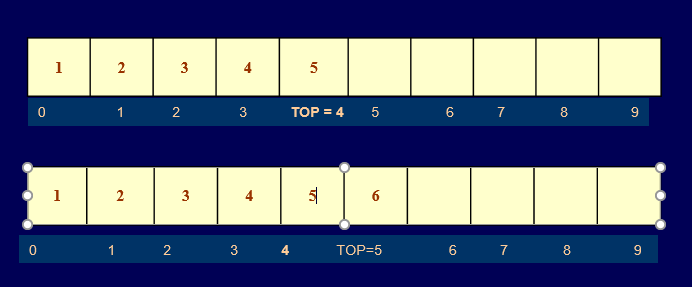
**Array Representation Of Stacks**

* In computer’s memory stacks can be represented as a linear array.
* Every stack has a variable TOP associated with it. TOP is used to store the address of the topmost element of the stack. It is this position from where the element will be added or deleted.
* There is another variable MAX which will be used to store the maximum number of elements that the stack can hold.
* If TOP = NULL, then it indicates that the stack is empty and if TOP = MAX -1, then the stack is full.



**Push Operation:**

* The push operation is used to insert an element into the stack. The new element is added at the topmost position of the stack. However, before inserting the value, we must first check if TOP=MAX-1, because if this is the case then it means the stack is full and no more insertions can further be done. If an attempt is made to insert a value in a stack that is already full, an OVERFLOW message is printed.



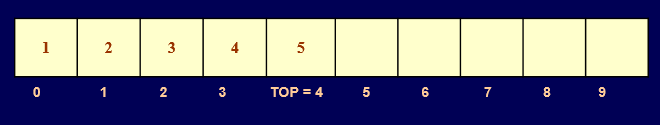
**POP OPERATION:**

* The pop operation is used to delete the topmost element from the stack. However, before deleting the value, we must first check if TOP=NULL, because if this is the case then it means the stack is empty so no more deletions can further be done. If an attempt is made to delete a value from a stack that is already empty, an UNDERFLOW message is printed.



**Peep Operation**

* Peep is an operation that returns the value of the topmost element of the stack without deleting it from the stack.
* However, the peep operation first checks if the stack is empty or contains some elements. For this, a condition is checked. If TOP = NULL, then an appropriate message is printed else the value is returned.

****

* Here Peep operation will return 5, as it is the value of the topmost element of the stack.

**Algorithm:**

**Algorithm For Push:**

PUSH(STACK,TOP,MAXSTK,ITEM)

This procedure pushes an ITEM on to a stack.

1.[Stack already filled?]

If TOP=MAXSTK, then Print: OVERFLOW, and Return.

2.Set TOP:=TOP+1.[Increases TOP by 1.]

3.Set STACK[TOP]:=ITEM.[Inserts ITEM in new TOP position.]

4.Return.

**Algorithm For Push:**

POP(STACK,TOP,ITEM)

This procedure deletes the top element of STACK and assigns it to the variable ITEM.

1.[Stack has an item to be removed?]

If TOP=0, then UNDERFLOW, and Return.

2.Set ITEM:=STACK[TOP]. [Assign TOP element to ITEM.]

3.Set TOP:=TOP-1.[Decreases TOP by 1.]

4.Return.

**Algorithm for Peep Operation**

**Step 1: IF TOP =NULL, then**

**PRINT “STACK IS EMPTY”**

**Go TO Step 3**

**Step 2: RETURN STACK[TOP]**

**Step 3: END**

**PROGRAM:**

//Write a program to perform Push, Pop, and Peek operations on a stack.

#include <stdio.h>

#include <conio.h>

#define MAX 5 // size of stack created

int st[MAX], top=-1;

void push(int st[], int val);

int pop(int st[]);

int peek(int st[]);

void display(int st[]);

//MAIN Function

void main() {

int val, option;

do

{

printf("\n \*\*\*\*\*MAIN MENU\*\*\*\*\*");

printf("\n 1. PUSH");

printf("\n 2. POP");

printf("\n 3. PEEK");

printf("\n 4. DISPLAY");

printf("\n 5. EXIT");

printf("\n Enter your option: ");

scanf("%d", &option);

switch(option)

{

case 1:

printf("\n Enter the number to be pushed on stack: ");

scanf("%d", &val);

push(st, val);

break;

case 2:

val = pop(st);

if(val != -1)

printf("\n The value deleted from stack is: %d", val);

break;

case 3:

val = peek(st);

if(val != -1)

printf("\n The value stored at top of stack is: %d", val);

break;

case 4:

display(st);

break;

}

}while(option != 5);

}

void push(int st[], int val)

{

if(top == MAX-1)

{

printf("\n STACK OVERFLOW");

}

else

{

top++;

st[top] = val;

}

}

int pop(int st[])

{

int val;

if(top == -1)

{

printf("\n STACK UNDERFLOW");

return -1;

}

else

{

val = st[top];

top--;

return val;

}

}

void display(int st[])

{

int i;

if(top == -1)

printf("\n STACK IS EMPTY");

else

{

for(i=top;i>=0;i--)

printf("\n %d",st[i]);

printf("\n"); // Added for formatting purposes

}

}

int peek(int st[])

{

if(top == -1)

{

printf("\n STACK IS EMPTY");

return -1;

}

else

return (st[top]);

}

**OUTPUT**

**CONCLUSION:**