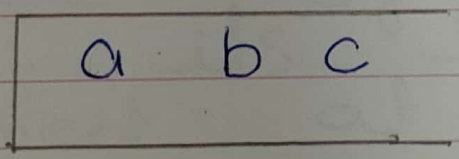
STACKS:

What is stack: A stack is an ordered list with the restrictions that elements are added or deleted from only one end of the list termed as **"top of the stack."** The other end of the list which lies within it is "inative" and called **"bottom of the stack."**

EXPLANATION:- IF S is a stack with three elements in it a, b, c where c occupies the top of the stack position, and if d were to be added, the resulted stack contents will

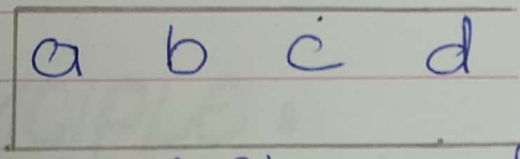
a, b, c, d. wher d occupies top of the stack position and for deleting operation, it would throw out the element at the top of the stack automatically. As the Fig illustration given bellow shows that.

Ordered list of element

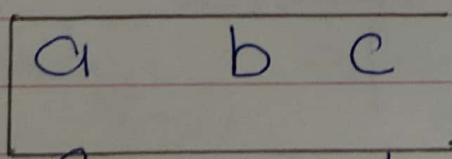


Bottom of stack

Top of stack



Addition of element d into the stack



Removel / deletion of element d from stack

The point should be kept in mind while dealing with this phenominum is:

1 During the insertion of element into stack it is essential that their identities are specified, whereas in the case of deletion this is not necessary since by by the virtue of its functionality, the element at the top of the stack position deleted automatically.

LIFO PRINCIPLE:

The stack data structure obeys the principle of LIFO which is the abbreviation of Last in first out. You can say element inserted into the stack

join last and that joined last are the first to be removed

STACK OPERATION:

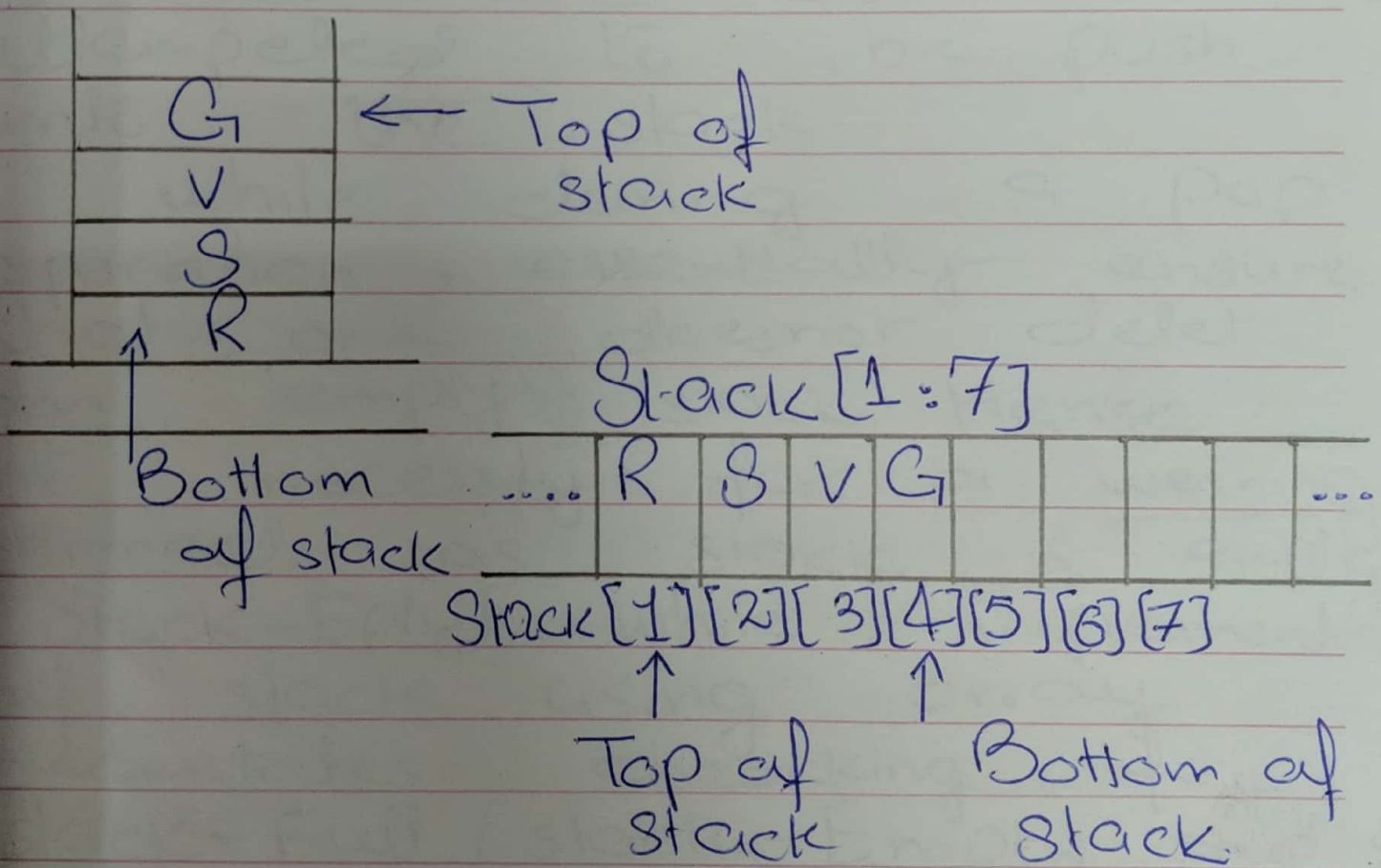
i PUSH: Insertion of elements
ii POP: Deletion of elements.

IMPLEMENTATION:

A common basic method of Implimentation is to mak use of the other fundamental data structure "Arrays". while arrays are siquential data structures the other of employing linked data structures have been successfully attempted and applied.

The fig shows an array base implimentation of stack. Considering the fact that stacks are imi-

dimentionaly ordered list and so arrays which despite their multi-dimentional structure are inherietently associated with a one-dimentional set of memory location.



This fig shows a stack of 4 elements R, S, V, G represented by an array stack [1:7]. Usually, if stack is represented

as array stack[1:n] then n elements can be store in it no more then n could be. So there is essential to issue a signa termed as STACK-FULL whe element whose number is over and above n are attempeted to be push into the stack.

while during a pop operation, essentially ensure that one doesnot delet an emply stack. Hence the necessity for a werning termed as stack is empty "Stack-Epty]. while implementa of stack using array necessitates cheecking for stack-Full / stack-Empty conditio during push/pop operations respectively, the implementatio of stack with linked data structure dispenses with these testing conditions

PUSH IMPLIMENTATION:

Let stack[1:N] an array implementation of stack and top be a varieble recording the currend top of stack position, top is initilize to 0. item is element to be add. n is maximum capacity.

Algorithm:

```
Push (STACK, n, top, item)
if (top=n) then STACK-FUL
else
    (top = top +1)
    Stack[top]= item.

end PUSH
```

POP IMPLEMENTATION:

In this case, as we know, no element identity need be specified since by default element is deleted. Here item is used as an output variable which stores a copy of element removed.

Algorithem:

```
Pop(stack, top, item)
if (top=0) then STACK EMPTY;
else { item = STACK[TOP];
       top = top - 1;

end top
```

Stack Applications:

1. Recursive Programing
2. Evaluation of Expression

Recursive Programing:

If a function call it self more then one time in its own body then this process is called recursion and this function is called recursive function.

Consider the recursive pseudo-code for factorial computation as shown below;

```
FACTORIAL (n)
if (n=0) then Factorial = 1;
else { x = n-1;
  X = FACTORIAL(X);
  FACTORIAL = n * Y;
end FACTORIAL
```

TAIL RECURSION: A speacl case of recursion where a recursive call to the function turns out to be the last action in calling function. But notice the recursive call needs to be the last executed statement in the functio not necessarily the last statemant

2: EVELUATING EXPRESSION:

INFIX, PREFIX and POST FIX

The evelucation of expression is an importent feacher of computer design. When we write or understand an arithmetic expression. — (A+B)↑C*D+E, we do so by following the scheme of (operand)(Opereto <operand> (i.e) an OPERATOR

is procedured and succeed by an <operand>. Such an expression is termed INFIX EXPRESION. It is already known how infix expression is used in programing, It is accorded rule of Hierarcy, Precedence and associcitivity to ensure that the computer does not misinterprete the expression but compute its value in a uniqueway

Actually a computer reworks on infix expression to produce an equivilent expression which follow the scheme of <operand><operand><operator> known as postfix. A third one catagory is one which follows the scheme of <operator><operand><operator>. and it is known as Prefix expression.

Page-1

```
importjava.util.*;
class stack
{
intstk[]=new int [4];
inttop,item;
Scanner in=new Scanner(System.in);
stack()
{
top=-1;
}
void push ()
{
if (top==3)
System.out.println("Stack is full");
else
{ top++;
System.out.print("Enter the number = ");
item=in.nextInt();
stk[top]=item;
}}
void pop ()
{
if (top==-1)
System.out.println("Stack is empty");
else
```

Page-2 Stack

```
{ item=stk[top];
top--;
System.out.println("item = "+item);

}}}
classapstack
{
public static void main(String arg [])
{ intch;
stack s=new stack();
Scanner in=new Scanner(System.in);
do
{
System.out.print("Enter the 1 to push and 2 to
pop = ");
ch=in.nextInt();
switch (ch)
{ case 1:
s.push();
break;
case 2:
s.pop();
break;
default:
System.out.print("wrong");}
```

Page-3

```
System.out.print("Enter the 0 to exit & any No
    to continue = ");
ch = in.nextInt();
while(ch != 0);
}}
```