Data Structures.

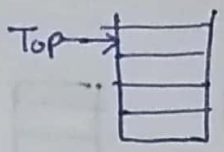
UNIT - 4

Stacks.

A stack is an ordered collection of homogeneous data elements, where the insertion & deletion takes place at one end, known as TOP.

The stack is also called as LAST IN FIRST OUT (LIFO)
It means the elements which is inserted last must be deleted first.

Eg. No. of plates in Cafeteria.
Stack of Coins



Basic Operations.

The basic operations are insertion, deletion, & Display.

PUSH:- Inserting or adding element in to Stack.
It places an object on the top of the stack

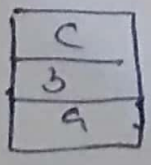
POP:- Deleting or removing element from the Stack
It removes an object from the top of stack

NOTE:- Is Empty:- Reports whether the stack is Empty or not

Is full:- Reports whether the stack exceeds limit or not

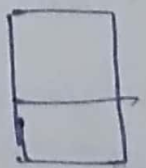
Stack

Push(S,a)
Push(S,b)
Push(S,c)



Push(S,d)

Stack overflow

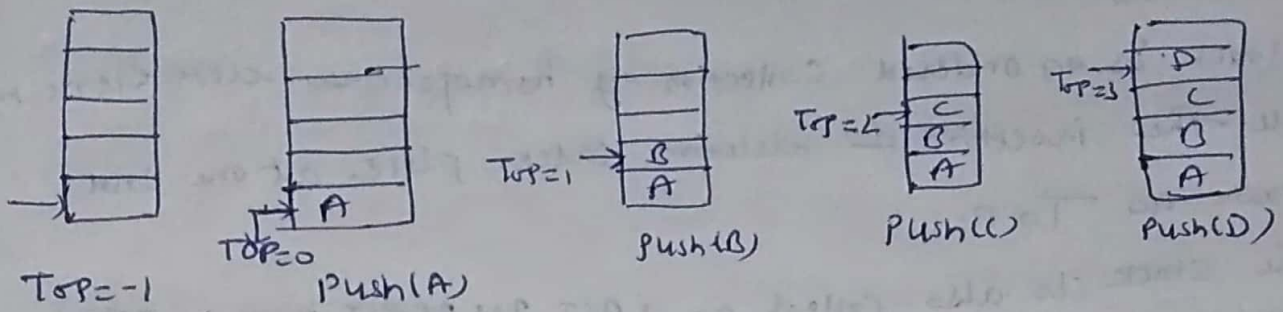


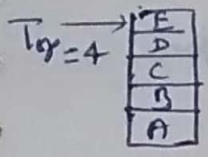
Pop(s).

Stack underflow → means stack is empty.

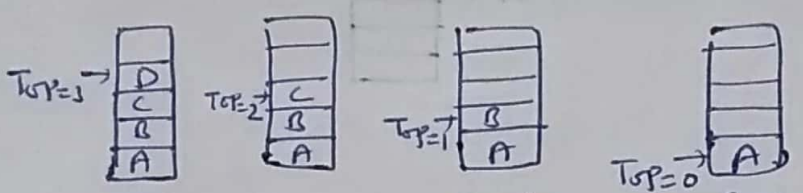
Eg:- Elements are inserted in the order as A, B, C, D, E

It represents the stack of 5 elements.





⇒ Now stack is full, if you want to delete element E has to be deleted first.



Abstract Data type for stack

ADT for stack

```
struct stack
    { int stack[5];
      int top;
    };
void push();  ← for inserting
void pop();   ← for deleting
void display(); ← for displaying the results.
```

Algorithm for inserting element into stack

Algorithm Push ()
1. If top = (size - 1)
   then write ('stack overflow')
   else
2. read item or data.
3. Top ← Top + 1
4. Stack [Top] ← item.
5. Stop.

Algorithm to delete element from stack

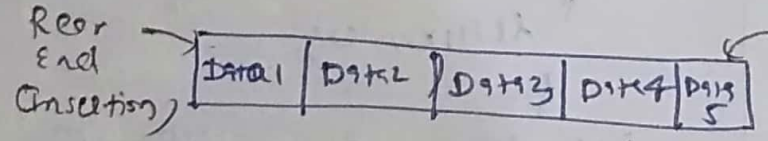
Algorithm Pop()
1. if top = -1
   then write ('Stack underflow')
   else
2. Item ← stack [Top]
3. Top ← Top - 1

Queues.

Queue is a linear list of elements in which deletion of an element can take place at one end, called **Front** and insertion can take place at other end, called **Rear**.

⇒ The first element in a queue will be first to be removed from the list.

⇒ Queue is also known as **FIFO (First in First Out)**



Application of Queue.

1. Serving requests on a single shared resources.
   Eg:- Printer, CPU task scheduling etc.
2. In real life, Call centre phone system (People should wait in hold with service representative is free)
3. Handling of interrupts in real time systems.

The Queue as an Abstract Data Type

A queue q of type T is a finite sequence of elements with the operations:

- MakeEmpty(q);- To make q as an empty queue
- IsEmpty(q)- To check q is empty, if empty return True or false.
- IsFull(q):- check q is full, if full return True or false
- Enqueue(q,x):- insert x at rear of the queue; if and only if and only if q is not full.
- Dequeue(q);- delete item from front of the queue, if q is not empty.
- Traverse(q):- To read entire queue that is display the content of the queue.

Implementation of Enqueue

```
int enqueue(int data)
{
  if(isfull())
  return 0;
  rear = rear+1;
  queue[rear]=data;
  return 1;
}
```

Implementation of Dequeue

```
int dequeue()
{
  if(isEmpty())
  { return 0;
  }
  int data = queue[front];
  front = front+1;
  return data;
}
```

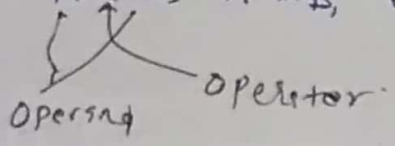
Application of Stack

1. Arithmetic Expression
   - (a) Infix notation
   - (b) Prefix notation or Polish notation.
   - (c) Postfix notation or reverse Polish Notation.
   - (d) Evaluation of Postfix Expression.
   - (2) Conversion of infix to Postfix Expression
   - (3) Conversion of infix to Prefix Expression.

(1) Infix Notation

The expression which is in Normal format, i.e. the operator lies b/w the operand.

eg. A+B, A*B, A/B etc.



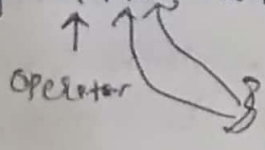
~~<operator>~~

<Operand><Operator><Operand>

(2) Prefix or Polish Notation

The operator comes first followed by the operands.

Eg. +AB, *AB, -AB, etc.

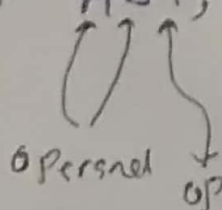


<Operator><Operands>

(3) Postfix or Reverse Polish Notation

The operand comes first followed by the Operator

Eg. AB+, CD-, AB/ etc.



<Operands> <Operator>

Infix to Postfix Conversion

Operands:- A, B, X, Y, P, Q, ---

Operators:- +, -, /, *, (,), ---

Rules:- (1) Priorities of operators.

^ ⇒ highest priority.

*/ ⇒ Next Priority (both have the same priority)

+,- ⇒ lowest Priority (" " " " " ")

(2) No two operators of same Priority can stay together in stack column.

(3) Lowest Priority can not be placed before highest priority.

(4) If any operator is present b/w the parenthesis just pop that operator.

Ex. (A+B/C*(D+E)-F).

Ans.

Symbol	Stack	Postfix
(	(	
A	(	A
+	(+	A
B	(+	AB
/	(+/	AB
C	(+/	ABC
*	(+*	ABC/
(	(+*(	ABC/
D	(+*(	ABC/D
+	(+*(+	ABC/D
E	(+*(+	ABC/DE

Note:- on going L to R in the given expn

If Operands come then operands will be placed in the postfix

If operator come then ~~pop that oper~~ put that operator in stack

Symbol	Stack	Postfix
)	(+* (+) ↑ Pop it	ABC/DE+
-	(+- ↑ Pop it	ABC/DE+*+
~~F~~	(+-	~~ABC/DE+*F~~
F	(-	ABC/DE+*+F
)	(-) ↑ Pop it	ABC/DE+*+F-

ABC/DE+*+F- Ans.

Infix to Prefix Conversion

Eg. A*B-C/D+E

Solution (A*B)-(C/D)+E

(*AB)-(/CD)+E

Let us Assume (*AB) = T1

(/CD) = T2

[T1 - T2] + E

(-T1T2) + E

Let us assume (-T1T2) = T3

∴ T3 + E

+T3E

+-T1T2E (Putting the value of T3)

+-*AB/CDE (Putting the value of T1 & T2)

Ans

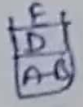
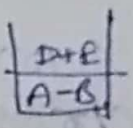
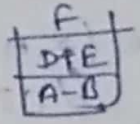
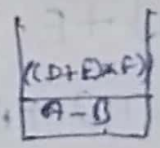
Q. Convert the Expression (A+B)/(C*D-E into prefix notation

Sol. → Practice this Question yourself

Conversion of Postfix Expn into Infix Expn

① AB- DE+f*/

Sol.

Reading of Postfix	Stack Top	Expression
1. A	A	
2. B	B	
3 −	(A-B)	
4. D	D	
5. E	E	
6. +	D+E	
7 f	f	
⑧ *	((D+E)*f)	

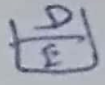
② / (A-B)/((D+E*F)/

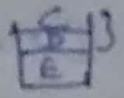
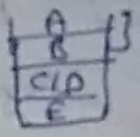
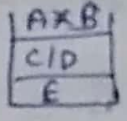
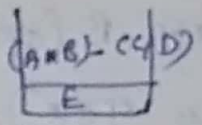
∴ Inf Expn is (A-B)/((D+E*F))

Conversion of prefix expn to Infix expn

Q. +- *AB/CDf

Sol. Reverse Prefix Expn fDC/BA*-+

Reading of prefix	Stack top	Expression
1. E	E	
2. D	D	

Reading of Prefix	Stack Top	Expression
3. C	C	
4 /	C	
	C/D	
5. B	B	
6. A	A	
7. *	A*	
8. −	(A*B)−(C/D)	
9. +	(A*B)−(C/D)+E	Ans.
10.)		

Evaluation of Postfix Expression

Suppose P is an arithmetic expression written in Postfix notation, the following algorithm, which uses a Stack to hold operands, evaluate P.

Algorithm:— Evaluate − Postfix (Stack, top, P)

// Stack:- It is a linear array.

// top:- top of the Stack, where top = −1

// P:- The given postfix expression

Step 1:- i = 0.

Step 2: while (P[i] != NULL)

(a) If (P[i] == Operand).

[PUSH Operand to the Stack].

(i) top = top+1

(ii) Stack [Top] = P[i].

[Endif].

(b) If (P[i] == Operator).

(i) [Pop 2 top most Operands.

(ii) Top = Top-1.

(iii) B = Stack[Top].

[Evaluate B (operator) A & put the result into the Stack 'Top'].

(iv) Stack [top] = B (Operator) A

[Endif].

(c) i = i+1

[End while].

Step 3:- [Display the Result].

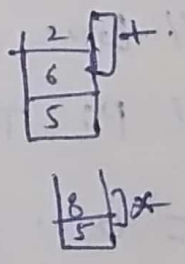
Print the Stack [TOP]

Step 4:- Exit

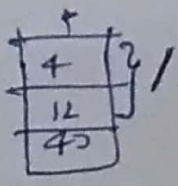
4. Evaluate the Postfix Notation.

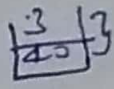
P: 5, 6, 2, +, *, 12, 4, /, -

Sol.	Symbol scanned	Stack
(1)	5	5
(2)	6	5, 6
(3)	2	5, 6, 2
(4)	+	5, 8
(5)	*	40
(6)	12	40, 12



Symbol scanned	Stack
(7) 4	40, 12, 4
(8) /	~~40, 12, 4,~~ 40, 3.
(9) -	37 Ans





Evaluation of Prefix Notation

Reverse the prefix Notation.

Q. Evaluate the Prefix Notation.

P: +, −, *, 2, 2, /, 16, 8, 5

Sol. Reverse the prefix Express

P: 5, 8, 16, /, 2, 2, *, −, +.

P = 5, 8, 16, /, 2, 2, *, −, +).

Symbol scanned	Stack
(1) 5	5
(2) 8	5, 8
(3) 16	5, 8, 16
(4) /	~~5, 8, 16,~~ 5, 2
(5) 2	5, 2, 2, 2
(6) *	~~5, 2, 2, 4~~ 5, 2, 4.
(7) −	5, 2
(8) +	7 ⇒ value
(9))