Sorting

Sorting is the process of arranging the elements of an array so that they can be placed either in ascending or descending order.

Sorting is of two types :-

1. Internal Sorting
2. External Sorting

1. Internal Sorting :-

When the entire collection of data is very small so that sorting can take place in the main memory of the computer, It is called Internal Sorting.

2. External Sorting :-

- External Sorting is a type of Sorting which can be implemented with the use of Secondary Storage.
- External Sorting is best when data is large enough that cannot be fit in the primary storage.

(2) Evaluation of Arithmetic Expressions :-

- An Arithmetic expression consists of operands and operators.
- In addition to operands and operators, the arithmetic expression may also include parenthesis like "left paren-thesis" and "right paren-thesis".

  Example :- A + (B - C)

- precedence Rules for Arithmetic Expression :-

Operators	Associativity	Precedence
^ exponentiation	Right to left	Highest followed by * Multiplication and / division.
* /	Left to Right	Highest followed by +, -
+ -	Left to Right	Lowest

Evaluation of Arithmetic Expressions requires two steps :-

(i) first convert the given expression into special notation.
(ii) Evaluate the expression in this new notation.

Notations for Arithmetic Expressions:-

(i) Infix Notation.
(ii) Prefix Notation.
(iii) Postfix Notation.

Infix Notation:-

The operator symbol is placed between the two operands.

A+B C*D E/F G-H

Example: A+B * C-D / F

We can also use the concepts of round brackets along with the operands and operator to specify what operation is to be performed first along with the precedence level.

(A+B) * (C-D)

Prefix notation (Polish Notation):-

It refers to the notation in which the operator

is placed before its two operands.

+AB * CD

We can easily convert the infix notation to the polish Notation, step by step, using the concept of square brackets [].

(A+B) * (C-D) = [+AB] * [-CD]

= *+AB-CD

Reverse Polish Notation (RPN)

→ In this type of expression operators comes after two operands.

AB+ CD *

→ It is also called Postfix Notation.

→ We can easily convert the infix Notation to the Postfix Notation, step by step using the concept of square bracket [].

(A+B) * (C-D)

= [AB+] * [CD-]

= AB+CD-*

* Evaluation of Postfix Expression:-

When an arithmetic expression is presented in the postfix form, you can use a stack to evaluate it to get value.

→ find the first operator from left to right and perform the operation.

→ It is free from any precedence.

Example:- a b c * -

Here first operator is * therefore b and c is multiplied first.

(b * c).

Subtract a and multiplied operand to get in final result.

Example:- 4 6 8 * -

Step 1:- (6 * 8)
Step 2:- 4 48 -
Step 3:- 48 - 4

Expression becomes 44

Algorithm:-

Start reading elements from left to right.

1. Read the next element.
2. If the element is operand then:-
   push element in stack.
3. If element is operator - then
   (a) pop two operands from stack
   (b) Evaluate (perform operation).
   (c) Push the result of expression into the stack top.
4. If no element then
   pop the result
   else
   go to step 1.
5. Stop.

for example:-

P = 4, 5, 2, +, *, 16, 4, /, -

We can use above algorithm to evaluate it.

P = 4, 5, 2, +, *, 16, 4, /, -

Symbol Scan	Stack
4	4
5	4, 5
2	4, 5, 2
+	4, 7
*	28
16	28, 16
4	28, 16, 4
/	28, 4
-	24
Result :-	24

(3) Conversion from INFIX to POSTFIX Notation :-

To convert infix expression to postfix expression we will use the stack data structure.

Input and Output

Input :-

The infix expression x ^ y / (5 * z) + 2

Output :-

Postfix form is :- xy ^ 5z * / 2 +

Algorithm:-

Step 1:- Push "(" on stack and ")" to end of infix expression

Step 2:- Scan infix expression from left to right and repeat following steps until stack is empty.

Step 3:- If an operand is found add to P. (Postfix).

Step 4:- If an operator is found, then:-

(a) Repeatedly pop all the operator from stack which are having higher or same precedence and put them into the P.

(b). Push the new operator to stack.

Step 5:- If left parenthesis is found push on stack.

Step 6:- If right parenthesis is found, then:-

(a) Repeatedly pop the stack and add the popped operators to the expression P until a left parenthesis is found.

(b) Remove the left parenthesis and don't add it in P.

Step 7:- Stop

Example:- Infix Notation (P/(Q-R) * S + T)

Symbol Scan	Stack	Postfix Expression
(	(	
P	(	P
/	(/	P
(	(/(	P
Q	(/(	PQ
-	(/(-	PQ
R	(/(-	PQR
)	(/	PQR-
*	(*	PQR-/
S	(*	PQR-/S
+	(+	PQR-/S*
T	(+	PQR-/S*T
)		PQR-/S*T+

Postfix Expression :- PQR-/S*T+

(4) Parenthesis Matching :-

- A bracket is considered to be any one of the following characters. (,), {, }, [, or].
- Two brackets are considered to be a matched pair if the an opening bracket (i.e. [, {, or (occurs to the left of a closing bracket (i.e.),], or }.

- Matched pairs of brackets : [], {}, and ().
- {[()]} is not balanced brackets. we can use a stack to store the unmatched brackets.

Algorithm:-

Scan the string from left to right, and for each char :-

1. If a left bracket, push onto stack.
2. If a right bracket, pop bracket from stack.
   (if not match or stack empty then fail).

Example:-

Initially :- Stack

Str	[	{	(	)	}	]

Opening bracket. Push into stack.

Step 1:- Stack

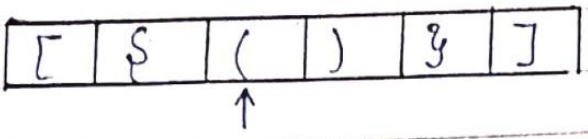
	[

Str	[	{	(	)	}	]

Opening bracket push into stack.

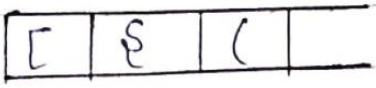
Step2:- Stack

Str

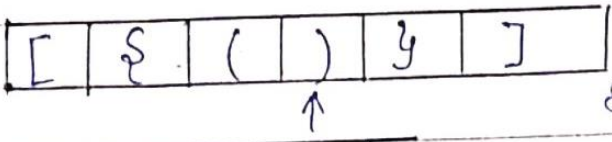


Opening bracket, push into stack.

Step3:- Stack

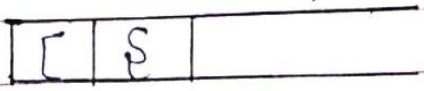


Str

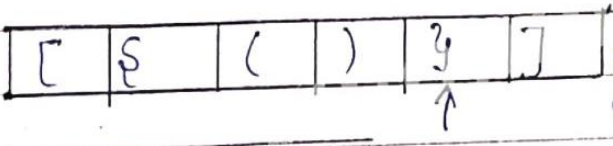


Closing bracket. Check top of the stack is same kind or not

Step4:- Stack



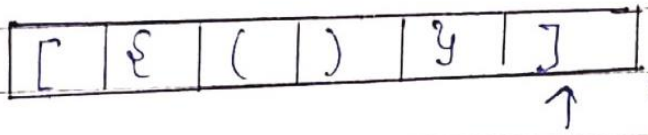
Str



Closing bracket check top of the stack is same kind or not.

Step 5:- Stack

Str



Closing bracket. Check top of stack is same kind or not.

(5) Recursion:-

- Recursion is a process of calling the function by itself and function is called recursive function.
- We can easily describe the implementation of recursion an application of stack using some of examples that use the concept of recursion.

Example of Recursion:-

1. factorial of positive integer.
2. Towers of Hanoi

1. Factorial of a Positive integer:-

factorial is a mathematical term.

factorial of a number, say n, is equal to the product of all integers from 1 to n.

factorial of n is denoted by:-

$$n! = 1 \times 2 \times 3 \ldots \times n.$$

$$5! = 5 \times 4 \times 3 \times 2 \times 1$$

$$fact(n) = \begin{cases} 1 & \text{if } n = 0 \text{ or } n = 1 \\ n * (n-1)! & \text{if } n > 1 \end{cases}$$

Algorithm:- factorial (N).

Step 1:- if N = 0 then
Return (1)
else
Return (N * factorial (N-1)
[end if]

Step 2:- End.

2. Towers of Hanoi :-

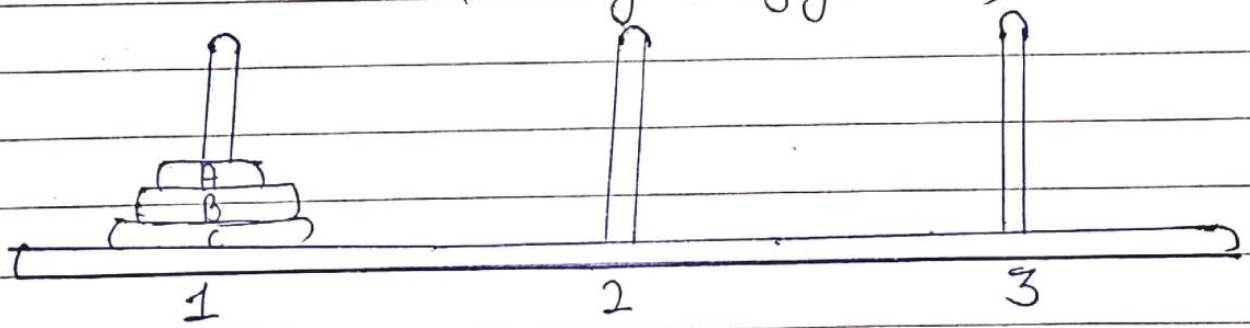
No discussion of recursion would be complete without the towers of Hanoi problem.

We are given a tower of N disks, initially stacked in increasing size on one of three poles.

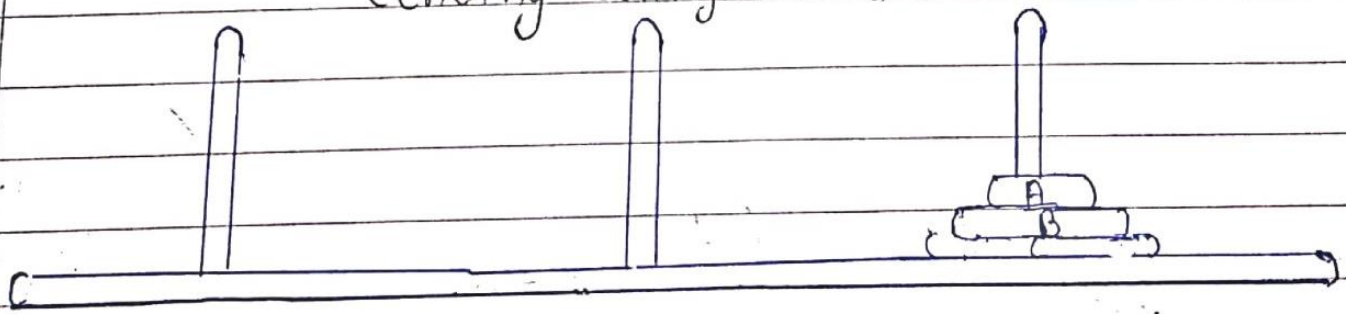
The objective is to transfer the entire tower to one of the other pole.

Rules :- (a) Move only one disc at a time.
(b) Only the top disc can be moved.
(c) A larger disc can not be placed on a smaller disc.

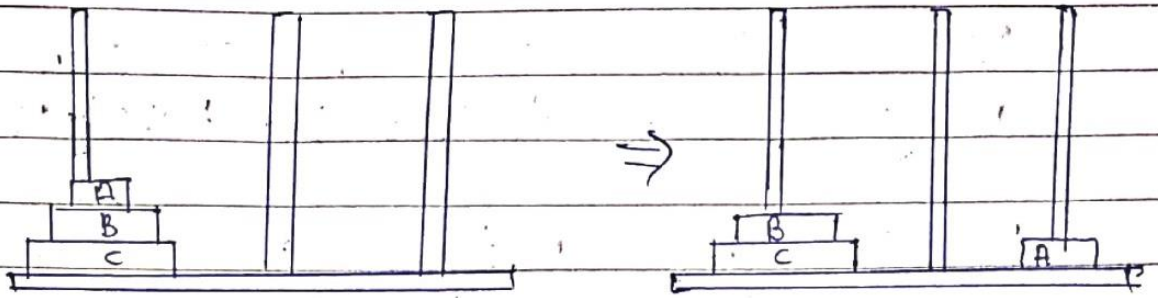
(Starting Configuration)



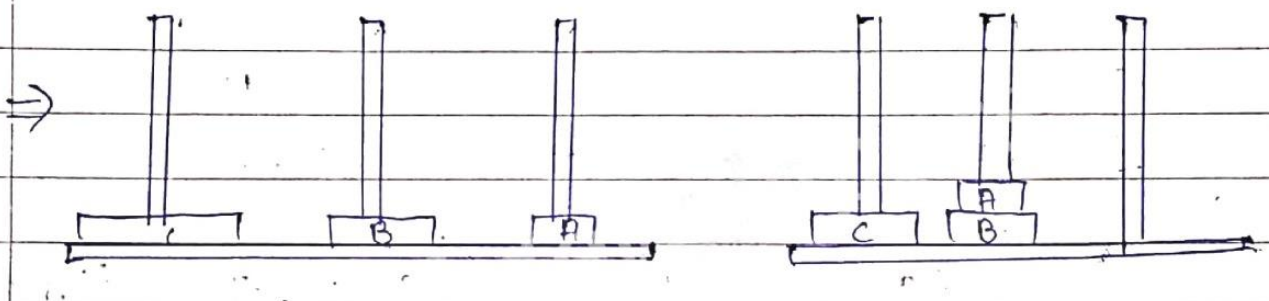
(Ending Configuration)



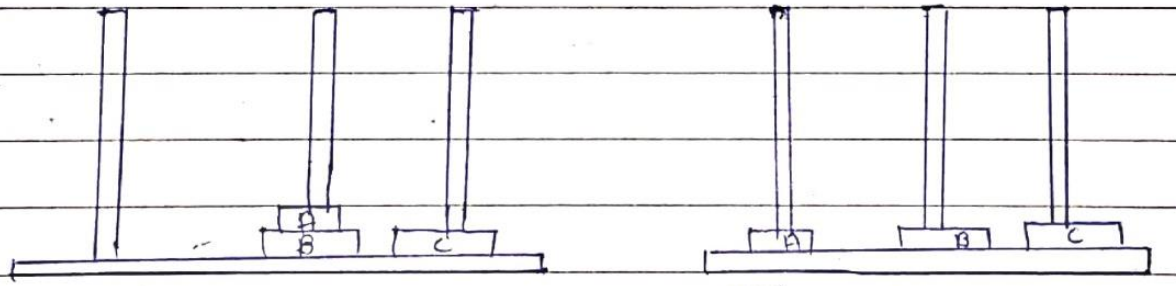
Solution:-



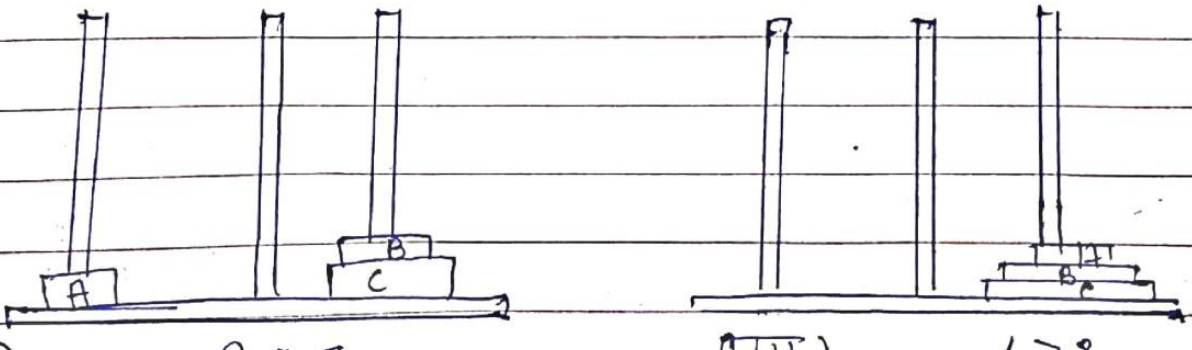
i) 1 → 3



(ii) 1 → 2 (iii) 3 → 2



(IV) 1 → 3 (V) 2 → 1



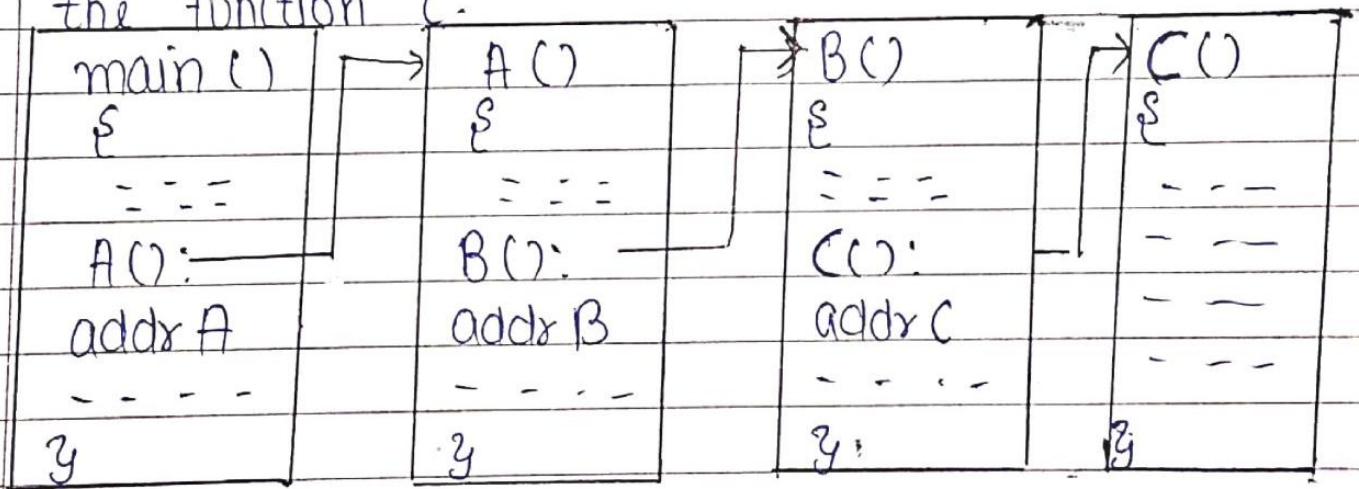
(VI) 2 → 3 (VII) 1 → 3

(6) Backtracking:-

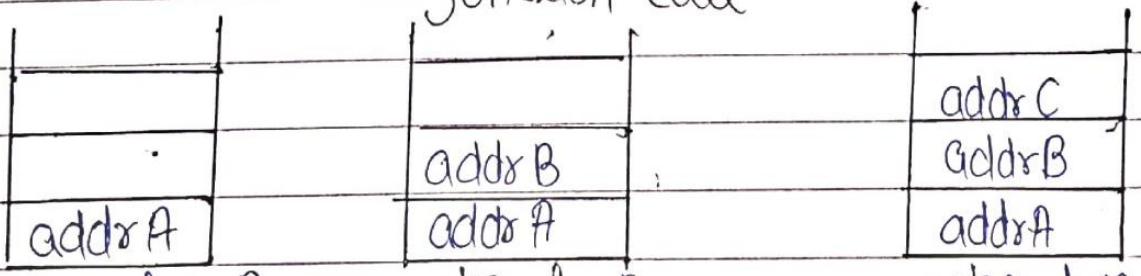
- Backtracking is used in algorithms in which there are steps along some path from some starting point to some goal.
- It is a recursive algorithm that is used for solving the optimization problem.

(7) Processing function calls:-

Suppose we have a program containing three functions :- A, B and C.
function A invokes function B, which invokes the function C.



function call



when fun. A is called | when fun B is called | when fun C is called