Applications of Stack

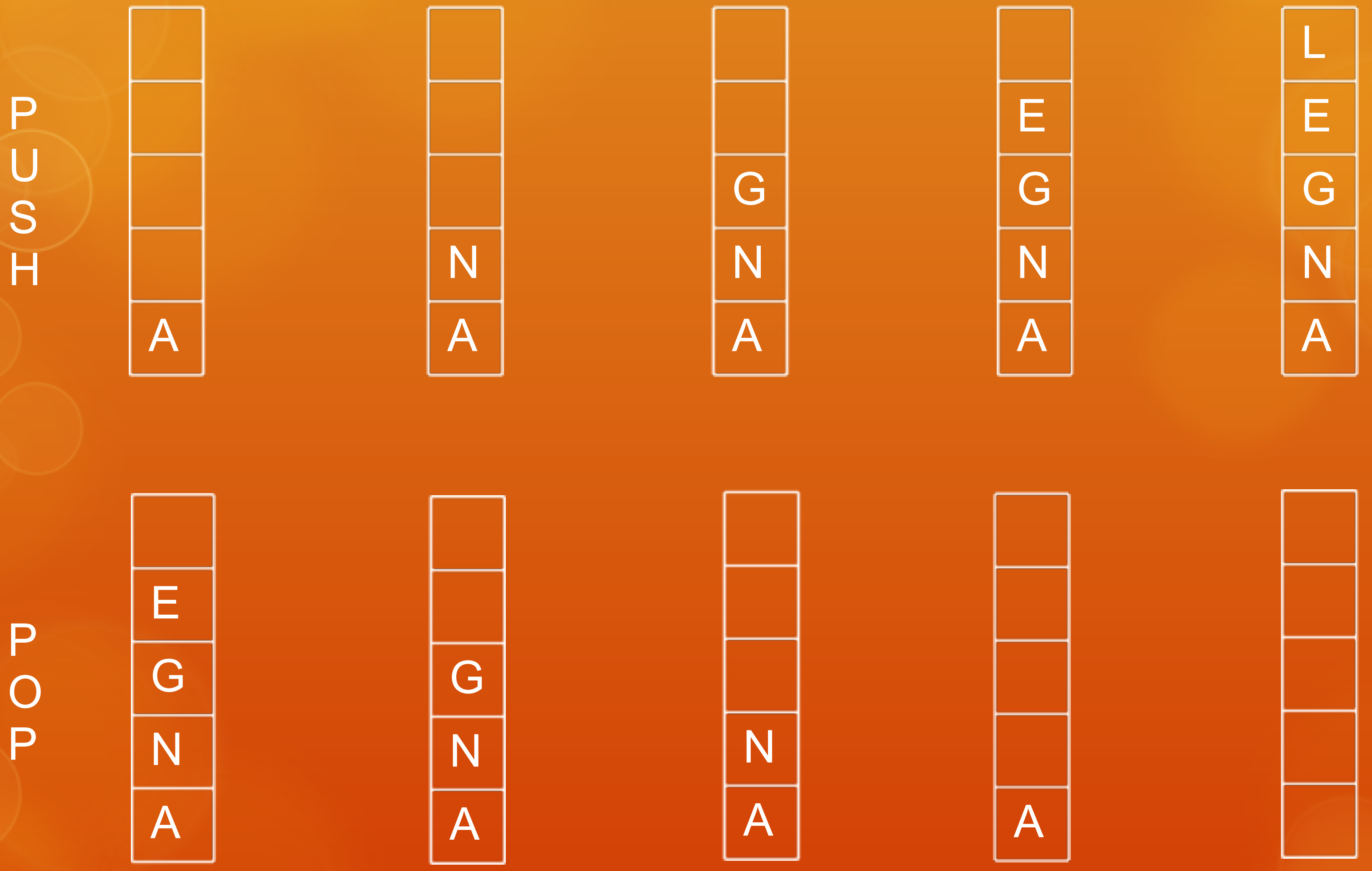
www.eshikshak.co.in

Reverse String or List

- We can accomplish this task by pushing each character or member from the string/list in the order it appears
- When the line is finished, characters are then proposed off the stack

“They come off in a reverse order”

Reverse string – Name="Angel"



Name : "L"

Name : "LE"

Name : "LEG"

Name : "LEGN"

Name : "LEGNA"

www.eshikshak.co.in

Polish Notation

- The process of writing the operators of expression either before their operands or after them is called “Polish Notation”
- The main property of Polish Notation is that the order in which operations are to be performed is ascertained by the position of the operators and operands in the expression

Polish Notation

- The notation refers to these complex arithmetic expressions in three forms:
  - If the operator symbols are placed between its operands, then the expression is in **infix notation**
    - **A + B**
  - If the operator symbols are placed before its operands, then the expression is in **prefix notation**
    - **+AB**
  - If the operator symbols are placed after its operands, then the expression is in **postfix notation**
    - **AB+**

Notation

Infix Notation	Prefix Notation	Postfix Notation
A + B	+ AB	AB +
(A - C) + B	+ - ACB	AC – B +
A + (B * C)	+ A * BC	ABC *+
(A+B)/(C-D)	/+ AB – CD	AB + CD -/
(A + (B * C))/(C – (D * B))	/+ A * BC – C * DB	ABC * + CDB * - /

Convert Infix expression to Postfix expression

A – It is an expression B – Postfix expression

Step 1. Push left parenthesis "(" into STACK and add right parenthesis ")" to the end of A

Step 2. Scan A from left to right and repeat step 3 to 6 for each element of A until the stack is empty

Step 3. If an operand is encountered, add it to B

Step 4. If a left parenthesis is encountered push it onto the stack

Step 5. If an operator is encountered then

1. Repeatedly pop from the STACK and add to B each operator (on the top of stack) which has the same precedence as or higher precedence than operator
2. Add operator to STACK

Step 6. If a right parenthesis is encountered, then

1. Repeatedly pop from the STACK and add to B each operator (on the top of STACK) until a left parenthesis is encountered
2. Remove the left parenthesis. (Do not add left parenthesis to B)

Step 7. Exit

Convert Infix expression to Prefix expression

- A – Arithmetic Expression B – Prefix Expression

Step 1. Push “)” onto STACK, and add “(“ to end of the A

Step 2. Scan A from right to left and repeat step 3 to 6 for each element of A until the STACK is empty

Step 3. If an operand is encountered add it to B

Step 4. If a right parenthesis is encountered push it onto STACK

Step 5. If an operator is encountered then:

a. Repeatedly pop from STACK and add to B each operator (on the top of STACK) which has

b. Add operator to STACK

Step 6. If left parenthesis is encontered then

a. Repeatedly pop from the STACK and add to B (each operator on top of stack until a left parenthesis is encounterd)

b. Remove the left parenthesis

Step 7. Exit

Evaluation of Postfix Expression

/* Reading the expression takes place from left to right */

Step 1. Read the next element /* first element for first time */

Step 2. If element is operand than

i. Push the element in the stack

Step 3. If element is operator then

i. Pop two operands from the stack /* POP one operand in case of NOT operator */

ii. Evaluate the expression formed by the two operands and the operator

iii. Push the results of the expression in the stack end.

Step 4. If no more elements then

i. POP the result

else

goto step 1

Step 5. Exit

Evaluation of Prefix Expression

/* Reading the expression take place from right to left /*

Step 1. Read the next element

Step 2. If element is operand then

i. Push the element in the stack

Step 3. If element is operator then

i. Pop two operands from the stack

ii. Evaluate the expression formed by two operands and the operator

iii. Push the results of the expression in the stack end

Step 4. if no more elements then

i. Pop the result

else

goto step 1

Step 5. Exit

Recursion

- The function which call itself (In function body) again and again is known as recursive function. This function will call itself as long as the condition is satisfied.
- This recursion can be removed by two ways:

    1. Through Iteration.

2. Through Stack

- Many mathematical functions can be defined recursively:
    - Factorial
    - Fibonacci
    - Euclid's GCD (greatest common denominator)
    - Fourier Transform

Factorial function in c

```
int fact(int n)
{
if(n==1)
{
return 1;
}
else
{
return(n*fact(n-1));
}
}
```

Use of STACK – FACTORIAL EXECUTION

n = 1	1	
n = 2	2 * fact(1)	2 * 1 = 2
n = 3	3 * fact(2)	3 * 2 = 6
n = 4	4 * fact(3)	4 * 6 = 24
n = 5	5 * fact(4)	5 * 24 = 120

Removal Through Stack

- Suppose P is a recursive procedure . The translation of recursive procedure P into a non recursive procedure work as follows:

- First of all, one defines:

1. A stack STPAR for each parameter PAR
2. A stack STVAR for each local variable VAR
3. A local variable ADD and a stack STADD to hold return address.

- The algorithm which translates the recursive procedure P into a non recursive procedure follows. It consist of three parts:

1. Preparation
2. Translating each recursive call P in procedure P.
3. Translating each return in procedure P.

1. Preparation

(a) define a stack STPAR for each parameter PAR, a stack STVAR for each local variable VAR, and a local variable ADD and a stack STADD to hold return address.

(b) Set TOP = NULL

2. Translation of “step K.call P.”

(a) Push the current values of the parameters and local variable onto the appropriate stacks, and push the new return address [step] K+1 onto STADD.

(b) reset the parameters using the new argument values.

(c) go to step 1.[The beginning of procedure P]

3. Translation of “Step J.return”

(a) if STADD is empty, then return.[control is return to the main program].

(b) Restore the top values of the stacks.That is, set the parameters and local variables equal to the top values on the stacks, and set ADD equal to the top value on the STADD.

(c) Go to the step ADD.

4. Step L. return

Translation of recursive to non-recursive procedure

- Declare STACK to hold all the local variables
- Push all the local variables and parameters called by value into the stack
- At the end of recursive function or whenever the function returns to the calling program, the following steps should be performed
    - If stack is empty, then the recursion has finished; make a normal return
    - Otherwise pop the stack to restore the values of all local variables and parameters called by value

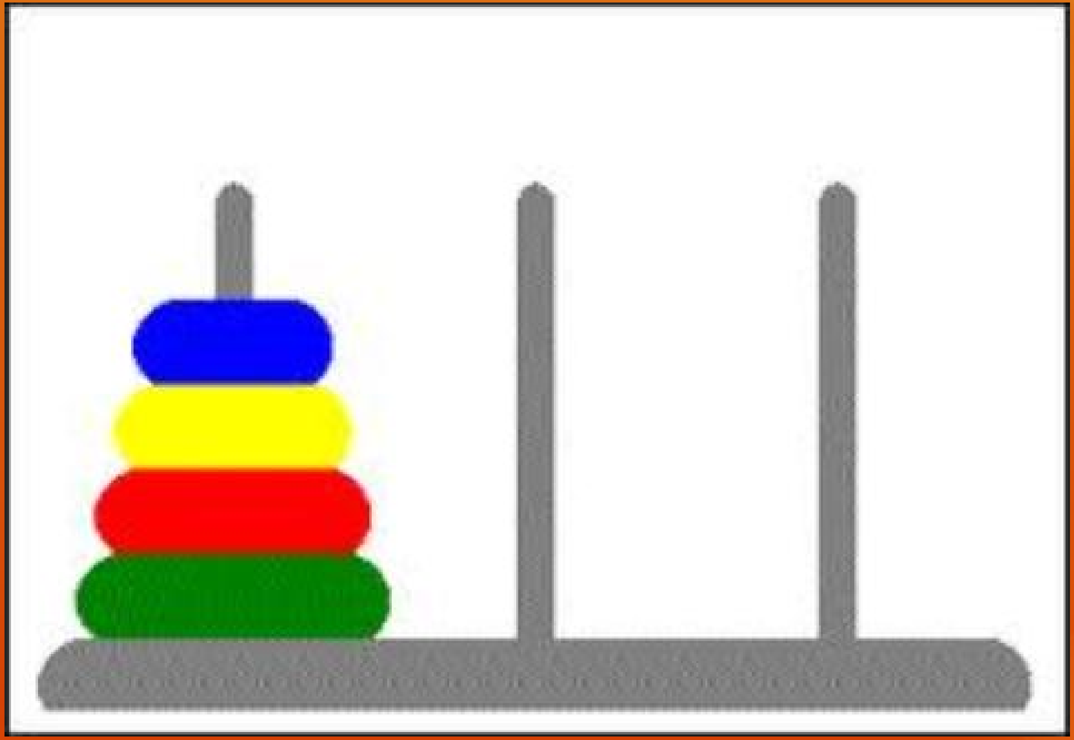
Difference between Recursion and Iteration

- ITERATION
  - It is a process of executing statements repeatedly, until some specific condition is specified
  - Iteration involves four clear cut steps, initialization, condition, execution and updating
  - Any recursive problem can be solved iteratively
  - Iterative computer part of a problem is more efficient in terms of memory utilization and execution speed

- RECURSIVE
  - Recursion is a technique of defining anything in terms of itself
  - There must be an exclusive if statement inside the recursive function specifying stopping condition
  - Not all problems has recursive solution
  - Recursion is generally a worst option to go for simple program or problems not recursive in nature

Tower of Hanoi

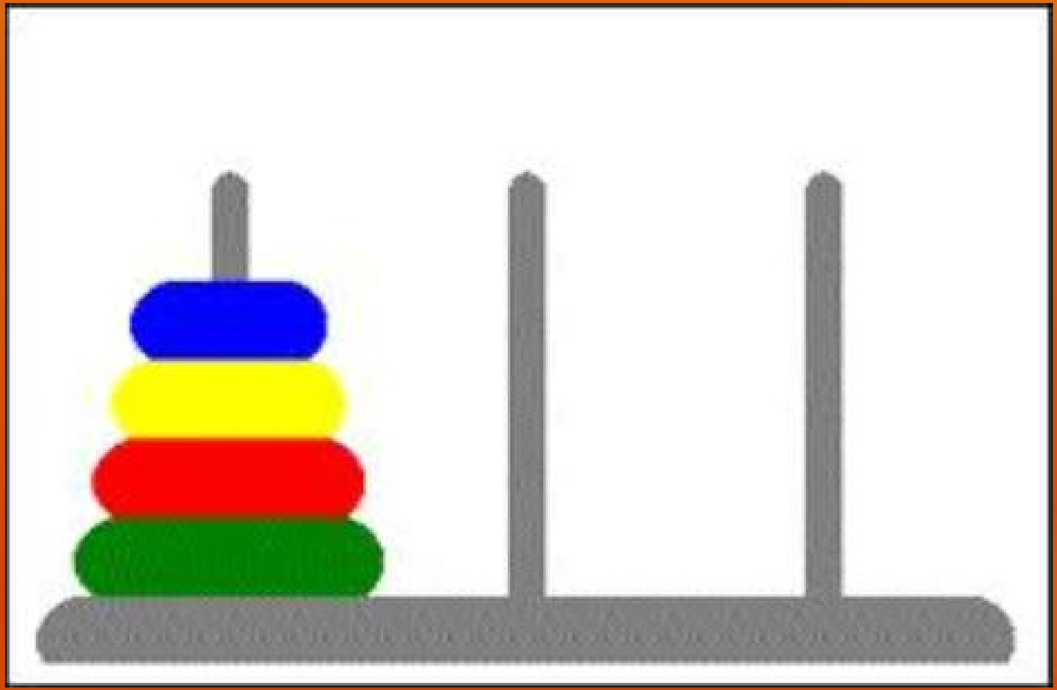
- Suppose three peg – A, B & C
- On peg A there are finite numbers of *n* disks with decreasing size



www.eshikshak.co.in

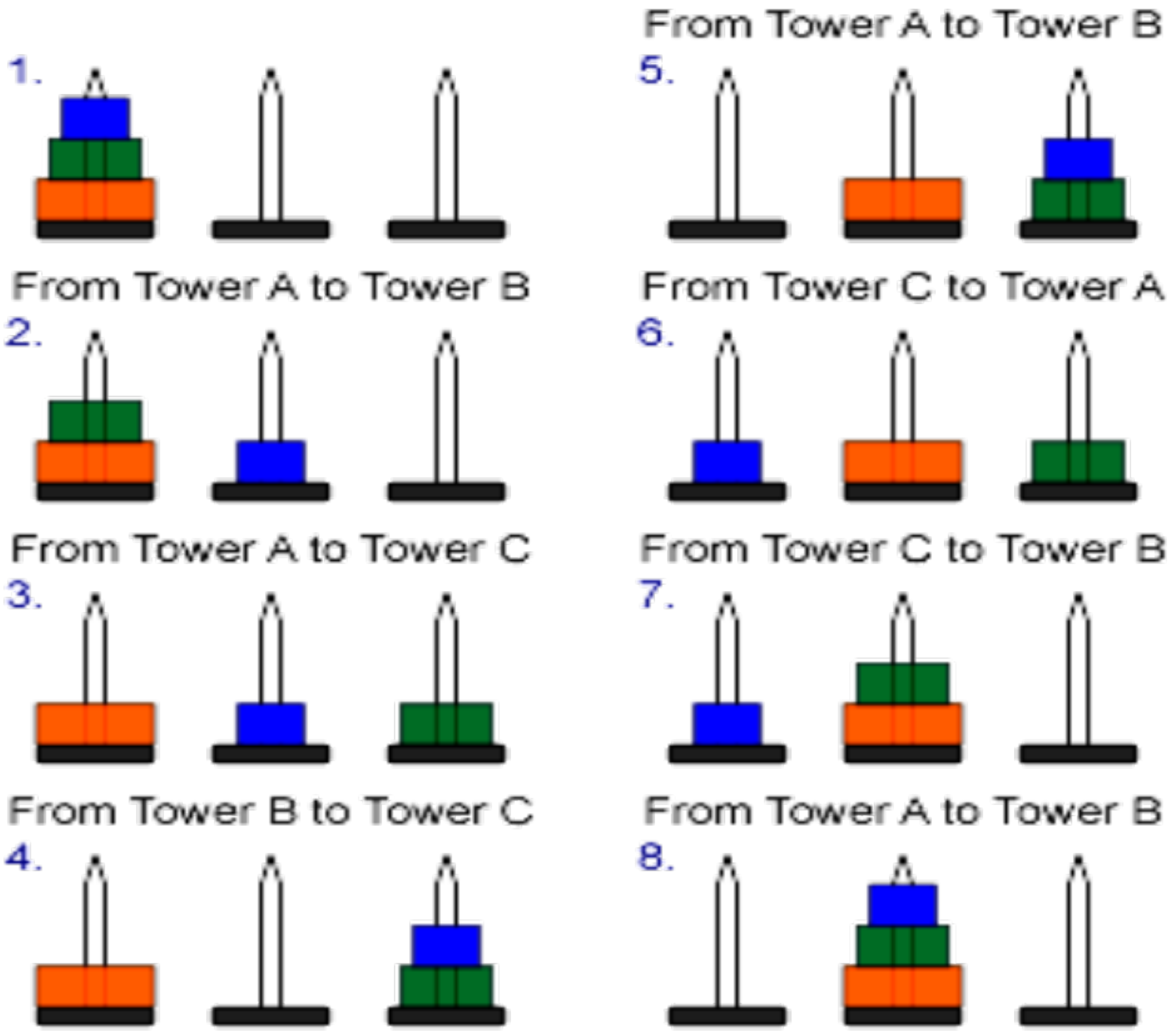
Tower of Hanoi - Rules

- Only one disk may be moved at a time (only the top disk on any peg may be moved to any other peg)
- A larger disk can be placed on a smaller (*smaller disk can not be placed on larger*)



www.eshikshak.co.in

Tower of Hanoi – Plates Move



Algorithm : Tower(N, BEG, AUX, END)

- This procedure gives a recursive solution to the Towers of Hanoi problem for N disks

```
Step 1. If N = 1 then
(a) Write : BEG -> END
(b) Return
End of If Structure
Step 2. [Move N – 1 disks peg BEG to peg AUX ]
Call TOWER(N-1, BEG, END, AUX)
Step 3. Write : BEG->END
Step 4. [Move N – 1 disks peg AUX to peg END]
Call TOWER(N-1, AUX, BEG, END)
Step 5. Return
```

www.eshikshak.co.in