MINOR PROJECT REPORT
ON
TOWER OF HANOI

in partial fulfilment for the award of the degree of

Bachelors of Computer Applications

SUBJECT – Data Structures

Submitted by:

Name: Karandeep

UID: 23BCA10034

Section: BCA – 3 "A"

Group: **1**st

Submitted to:

Name: Ms. Shilpi Mittal

Designation: Co-ordinator

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

ABSTRACT

Introduction:

Tower of Hanoi

The Tower of Hanoi is a mathematical puzzle. There are three pegs, source(A), Auxiliary (B) and Destination(C). Peg A contains a set of disks stacked to resemble a tower, with the largest disk at the bottom and the smallest disk at the top. Below Fig - 1 Illustrate the initial configuration of the pegs for 3 disks. The objective is to transfer the entire tower of disks in peg A to peg C maintaining the same order of the disks.

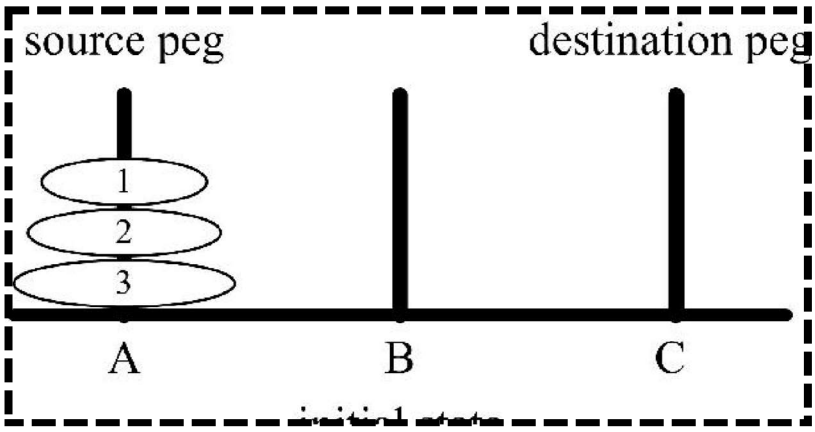

Fig - 1

Technique:

The program employs a recursive algorithm, if statement:

Recursive case:

- Move n-1 disks to an auxiliary peg.
- Move the nth disk to the destination peg.
- Move the n-1 disks from the auxiliary peg to the destination.

If statement:

The condition if (n == 1) identifies the simplest scenario—when there is only one disk to move.

If the condition is true, the program prints the move for that single disk and then exits the function using return, preventing further recursive calls.

Operating System:

S/W:

- Visual studio or Block Linux and macOS with a suitable C++ compiler like g++.

H/W:

- **Processor**: A modern CPU (Intel or AMD) with a minimum of 1 GHz.
- **RAM**: At least 2 GB of RAM, although 4 GB or more is recommended for smooth performance.
- **Graphics**: Integrated graphics are sufficient; no high-end graphics card is required.
- **Storage**: At least 1 GB of free disk space to install the IDE and any necessary compilers.

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

SUMMARY

Input:

The program prompts the user to enter the number of disks (n) that will be moved between the pegs.

Process:

The function TOH (int n, char Sour, char Aux, char Des) recursively solves the problem:

- Base Case: If there is only 1 disk (n == 1), it directly moves the disk from Sour to Des and returns.
- Recursive Case:

    1. Move n-1 disks from Sour to Aux using Des as an auxiliary.
    2. Move the nth disk from Sour to Des.
    3. Move the n-1 disks from Aux to Des using Sour as an auxiliary.

Output:

Suppose the user inputs n = 3. The recursive steps and output will be:

1. Move 2 disks from A (Source) to B (Auxiliary) using C (Destination).
2. Move 1 disk from A to C.
3. Move 2 disks from B to C using A