TITLE OF PROJECT

Production System for Tower of Hanoi

END TERM REPORT

by

CANDIDATE(S)

Reg Number	Name	Sec / Roll No.
11814881	DEEPAK KUMAR MAKHIJA	65
11814867	SHUBHAM VERMA	58

Department of Intelligent Systems
School of Computer Science Engineering
Lovely Professional University, Jalandhar
April – 2020

Student Declaration

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we are shall take full responsibility for it.

Name: Deepak k. Makhija

: Shubham Verma

Place –HOME
Date –

TABLE OF CONTENTS

BONAFIDE CERTIFICATE

Certified that this project report “**Production System for solving Towers of Hanoi**” is the bonafide work of “ **SHUBHAM VERMA** AND **DEEPAK KUMAR MAKHIJA**” who carried out the project work under my supervision.

<<Signature of the Supervisor>>

Department of Intelligent Systems

Background and objectives :

The tower of Hanoi (commonly also known as the "*towers* of Hanoi"), is a game invented by E. Lucas in 1883. It is also known as the Tower of Brahma puzzle and appeared as an intelligence test for apes in the film *Rise of the Planet of the Apes* (2011) under the name "Lucas Tower."

Given a stack of n disks arranged from largest on the bottom to smallest on top placed on a rod, together with two empty rods, the tower of Hanoi puzzle asks for the minimum number of moves required to move the stack from one rod to another, where moves are allowed only if they place smaller disks on top of larger disks. The puzzle with $n=4$ pegs and n disks is sometimes known as Reve's puzzle.

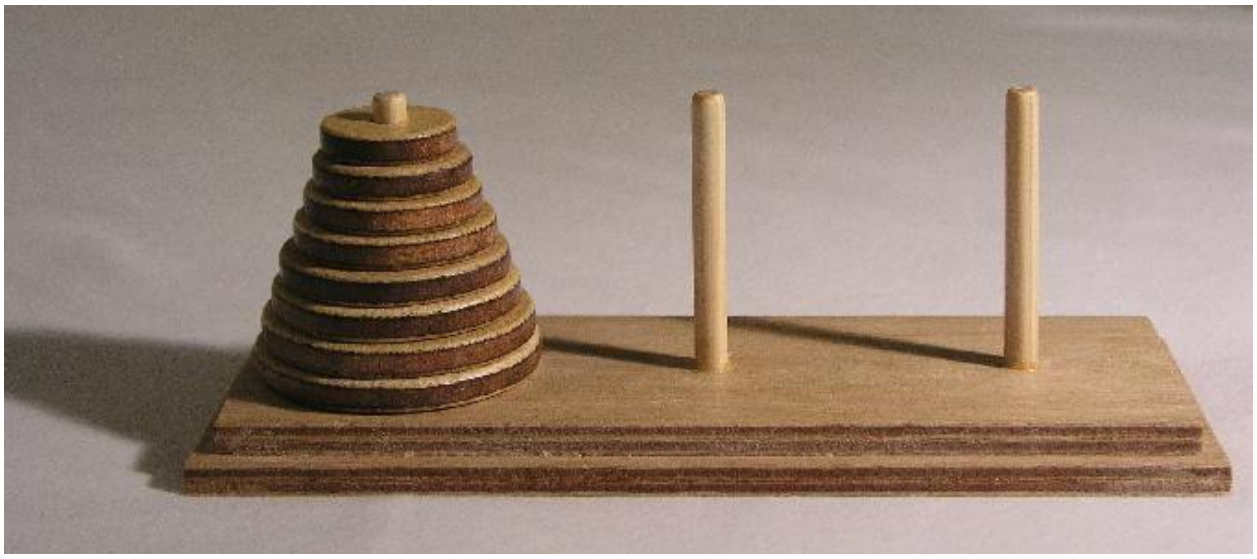
The mission is to move all the disks to some another tower without violating the sequence of arrangement. A few rules to be followed for Tower of Hanoi are −

- Only one disk can be moved among the towers at any given time.
- Only the "top" disk can be removed.
- No large disk can sit over a small disk.

With 3 disks, the game can be solved in least 7 moves. The minimal number of moves required to solve a Tower of Hanoi puzzle is $2^n - 1$, where *n* is the number of disks.

Description of Project :

Tower of Hanoi for 8 disks:



To write an algorithm for Tower of Hanoi, first we need to learn how to solve this problem with lesser amount of disks, say → 1 or 2. We mark three towers with name, **source**, **destination** and **aux** (only to help moving the disks). If we have only one disk, then it can easily be moved from source to destination peg.

If we have 2 disks −

- First, we move the smaller (top) disk to aux peg.
- Then, we move the larger (bottom) disk to destination peg.
- And finally, we move the smaller disk from aux to destination peg.

So now, we are in a position to design an algorithm for Tower of Hanoi with more than two disks. We divide the stack of disks in two parts. The largest disk (n^{th} disk) is in one part and all other (n-1) disks are in the second part.

Our ultimate aim is to move disk **n** from source to destination and then put all other (n1) disks onto it. We can imagine to apply the same in a recursive way for all given set of disks.

Algorithm used:

In this problem Recursion is the method used by which this problem is divided into sub-sol which then by backtracking gives the output.'

Here depth for search is used as first we to till the extreme state which is the beginning of sol and and by backtracking the output is generated.

1. Move m − 1 disks from the source to the spare peg, by the same general solving procedure. Rules are not violated, by assumption. This leaves the disk m as a top disk on the source peg.
2. Move the disk m from the source to the target peg, which is guaranteed to be a valid move, by the assumptions — a simple step.
3. Move the m − 1 disks that we have just placed on the spare, from the spare to the target peg by the same general solving procedure, so they are placed on top of the disk m without violating the rules.
4. The base case being to move 0 disks (in steps 1 and 3), that is, do nothing – which obviously doesn't violate the rules.

In code:

- Label the pegs A, B, C,
- Let *n* be the total number of disks,
- Number the disks from 1 (smallest, topmost) to *n* (largest, bottom-most).

- **Monotonic Production System**: It's a production system in which the application of a rule never prevents the later application of another rule, that could have also been applied at the time the first rule was selected. As we are also backtracking in this problem so it is monotonic.

- **Partially Commutative Production System**: It's a type of production system in which the application of a sequence of rules transforms state X into state Y, then any permutation of those rules that is allowable also transforms state x into state Y. Theorem proving falls under the monotonic partially communicative system. As we know to goal state and only one goal state can be there with diff paths to reach there.

7 Characteristic of AI :

Decomposable to smaller or easier problems:

Here in this problem as recursion is used so each movement of disk is taken as sub-problem which goes till end of the solution and by backtracking outputs are generated

Solution steps can be ignored or undone:

Here in this problem as recursion is used every movement of disk creates a sub-solution and some of this solution can be valid and some can be invalid. But in this problem only valid sub-sol are displayed.

Predictable problem universe:

As this problem has a certain outcome which we know just we need is to reach to that problem with some steps based on some conditions so this problem is universaly predictable.

Is a good solution absolute or relative:

It is a absolute solution as we know the goal state. As only one solution is possible but path can be different.

Is a solution a state of path:

This solution is a path to goal state as in this question a path is followed to reach a goal state.

Requires lots of knowledge or uses knowledge to constrain solutions:

As there are some conditions which are used as knowledge to solve this problem. Conditions like one disk can only be moved at a time and no bigger disk can be kept over a smaller disk.

Requires periodic interaction between human and computer:

This problem is interactive to some extend as we need to tell computer the no. of disk for which we need to solve the towe of Hanoi but after it we just get the output without any further inputs.

Description of Work Division :

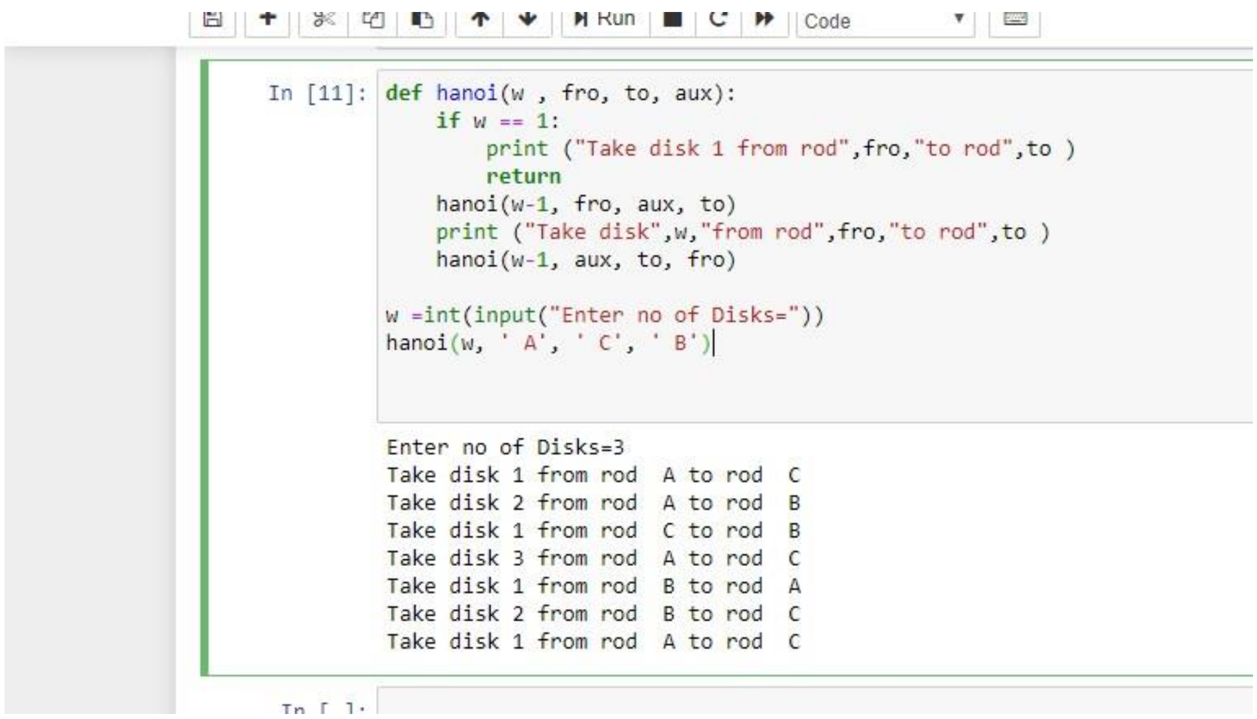
Reg No.	Name	Work
11814867	SHUBHAM VERMA	Background and Objectives, some report, Description of Project, some report
11814881	DEEPAK KUMAR MAKHIJA	Python Code for tower of Hanoi, Technologies , Framework and SWOT Analysis

Implementation :

Code:
```
def hanoi(w , from,
to, aux):
   if q == 1:         print ("Take disk 1 from rod
",from,"to rod ",to )         return
   hanoi(w-1, from, aux, to)
   print ("Take disk",w,"from rod ",from,"to rod ",to )
hanoi(w-1, aux, to, from)

w =int(input("Enter no of Disks=")) hanoi(w,
' A', ' C', ' B')
```

For 3 disks :

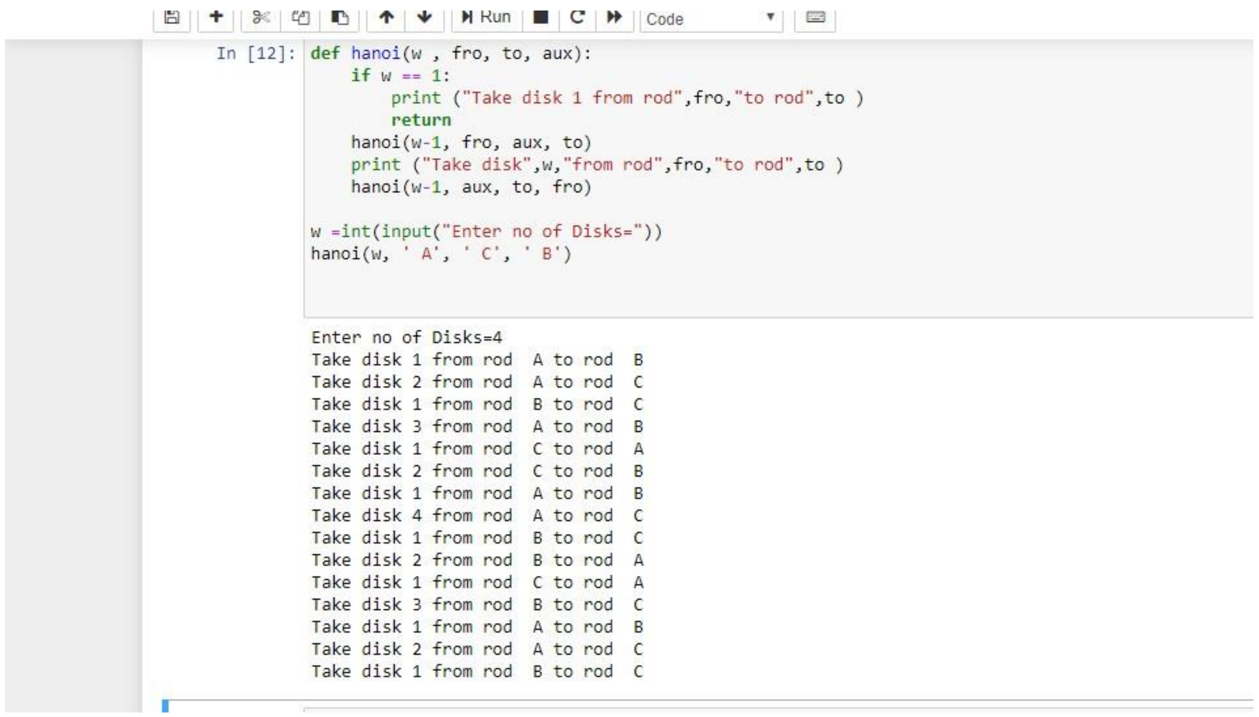


```
In [11]: def hanoi(w , fro, to, aux):
             if w == 1:
                 print ("Take disk 1 from rod",fro,"to rod",to )
                 return
             hanoi(w-1, fro, aux, to)
             print ("Take disk",w,"from rod",fro,"to rod",to )
             hanoi(w-1, aux, to, fro)

         w =int(input("Enter no of Disks="))
         hanoi(w, ' A', ' C', ' B')

         Enter no of Disks=3
         Take disk 1 from rod  A to rod  C
         Take disk 2 from rod  A to rod  B
         Take disk 1 from rod  C to rod  B
         Take disk 3 from rod  A to rod  C
         Take disk 1 from rod  B to rod  A
         Take disk 2 from rod  B to rod  C
         Take disk 1 from rod  A to rod  C
```

For 4 disks:



```
In [12]: def hanoi(w , fro, to, aux):
             if w == 1:
                 print ("Take disk 1 from rod",fro,"to rod",to )
                 return
             hanoi(w-1, fro, aux, to)
             print ("Take disk",w,"from rod",fro,"to rod",to )
             hanoi(w-1, aux, to, fro)

         w =int(input("Enter no of Disks="))
         hanoi(w, ' A', ' C', ' B')

         Enter no of Disks=4
         Take disk 1 from rod  A to rod  B
         Take disk 2 from rod  A to rod  C
         Take disk 1 from rod  B to rod  C
         Take disk 3 from rod  A to rod  B
         Take disk 1 from rod  C to rod  A
         Take disk 2 from rod  C to rod  B
         Take disk 1 from rod  A to rod  B
         Take disk 4 from rod  A to rod  C
         Take disk 1 from rod  B to rod  C
         Take disk 2 from rod  B to rod  A
         Take disk 1 from rod  C to rod  A
         Take disk 3 from rod  B to rod  C
         Take disk 1 from rod  A to rod  B
         Take disk 2 from rod  A to rod  C
         Take disk 1 from rod  B to rod  C
```

Technologies and Framework used :

We used Python Language to implement the code.

We used Jupyter as a platform to write and run code which works under Anaconda3 which is easy to use.

Recursion is the algorithm used.