**REPORT ON**

**“The Tower Of Hanoi”**

# SUBMITTED BY

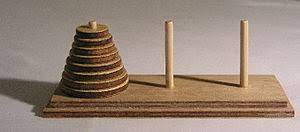
SAYYED FATIMA TAIBA MOHD HANIF

**INDEX**

**INTRODUCTION**

**TOWER OF HANOI**

The Tower of Hanoi is a mathematical game or puzzle consisting of three rods and a number of disks of various diameters, which can slide onto any rod. The puzzle begins with the disks stacked on one rod in order of decreasing size, the smallest at the top, thus approximating a conical shape.



**OBJECTIVE OF TOWER OF HANOI**

* The objective of the puzzle is to move the entire stack to another rod, obeying the following rules:
* Only one disk can be moved at a time.
* Each move consists of taking the upper disk from one of the rods and sliding it onto another rod, on top of the other disks that may already be present on that rod.
* No disk can be placed on top of a smaller disk than itself.

**OBJECTIVE OF THIS PROJECT**

* In this project our main goal is to take number of disks from the user to show how the concept of tower of Hanoi works.
* In this project we show how the disks replace from the first tower to the 3rd tower without placing the larger disks over het smaller disks.
* We have used recursion formula for indicating the placements.
* We have used recursion formula for solving tower of Hanoi

void TowerOfHonai(char sourceTower, char tempTower, char targetTower, int n)

{

    if(n>0)

    {

    TowerOfHonai(sourceTower, targetTower,tempTower,n-1);

    //cout<<"MOVE DISK "<<n<<" FROM "<<sourceTower<<" TO "<<targetTower<<endl;

    TransferDisk(sourceTower, targetTower, n);

    TowerOfHonai(tempTower, sourceTower,targetTower,n-1);

}

}

**THE FUNCTIONS**

void drawPole()

{

      setcolor(LIGHTGREEN) ;

      line(10,452,600,452) ;

      line(100,452,100,100) ;

      line(300,452,300,100) ;

      line(500,452,500,100) ;

}

void drawBoard()

{

      setcolor(RED);

      line(1,1,610,1);

      line(1,1,1,480);

      line(610,1,610,480);

      line(1,479,610,479);

      settextstyle(TRIPLEX\_FONT,HORIZ\_DIR,4) ;

      setcolor(YELLOW) ;

      outtextxy(90,10,"L") ;

      outtextxy(290,10,"C") ;

      outtextxy(490,10,"R") ;

}

void InitializeScreen()

{

      int gDriver=DETECT,gMode ;

      initgraph(&gDriver,&gMode,"C:\\TURBOC3\\BGI") ;

      drawPole();

      drawBoard();

      /\*

      getch();

      setcolor(MAGENTA) ;

      rectangle(50,100,150,110) ;

      getch();

      setcolor(BLACK) ;

      rectangle(50,100,150,110) ;

      getch();

      setcolor(MAGENTA) ;

      rectangle(250,100,350,110) ;

      getch();   \*/

}

void InitializeDisk(Disk\* myDisk[],int n)

{

    int i=0;

    int j=0;

    for(i=n,j=0; i>0; i--,j++)

    {

          myDisk[i] = new Disk(100, 450-10 - j\*20, 10+i\*20, 18);

          myDisk[i]->DrawDisk();

          getch();

    }

    LeftDiskCount=n;

    CenterDiskCount=0;

    RightDiskCount=0;

    TotalDisk=n;

}

void TransferDisk(char sourceTower, char targetTower, int n)

{

    int newX = 0;

    int newY = 0;

    myDisks[n]->EraseDisk();

    getch();

    switch(sourceTower)

    {

        case 'L':

               LeftDiskCount--;

               break;

        case 'C':

               CenterDiskCount--;

               break;

        case 'R':

               RightDiskCount--;

               break;

    }

    switch(targetTower)

    {

        case 'L':

               LeftDiskCount++;

               newX=100;

               newY=450-10 - (LeftDiskCount-1)\*20;

               break;

        case 'C':

               CenterDiskCount++;

               newX=300;

               newY=450-10 - (CenterDiskCount-1)\*20;

               break;

        case 'R':

               RightDiskCount++;

               newX=500;

               newY=450-10 - (RightDiskCount-1)\*20;

               break;

    }

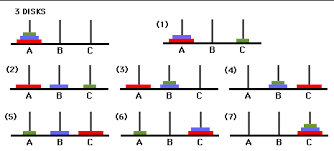
    myDisks[n]->setPosition(newX,newY);

    myDisks[n]->DrawDisk();

    getch();

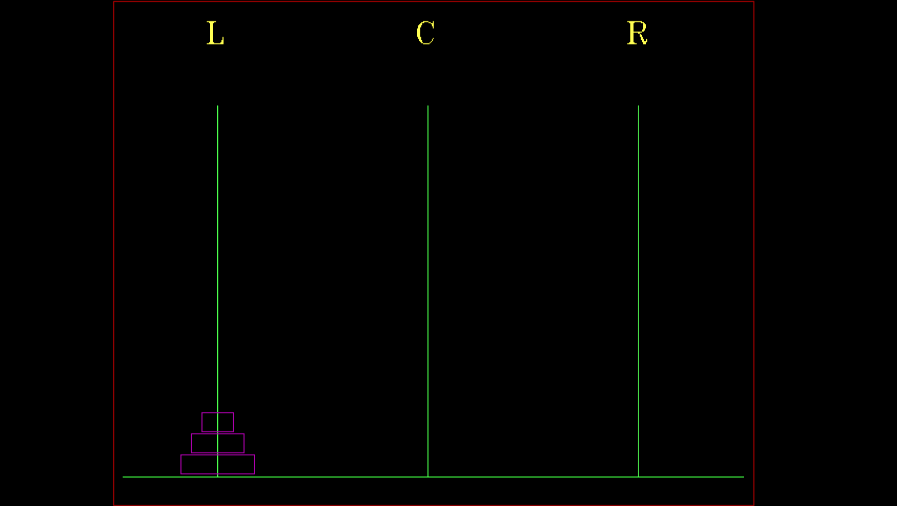
}

ILLUSTRATION OF A 3 DISK TOWER OF HANOI

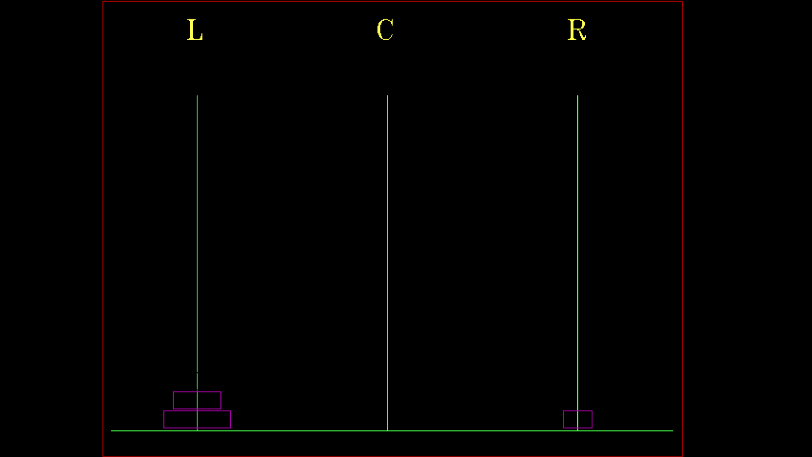


**OUTPUT**

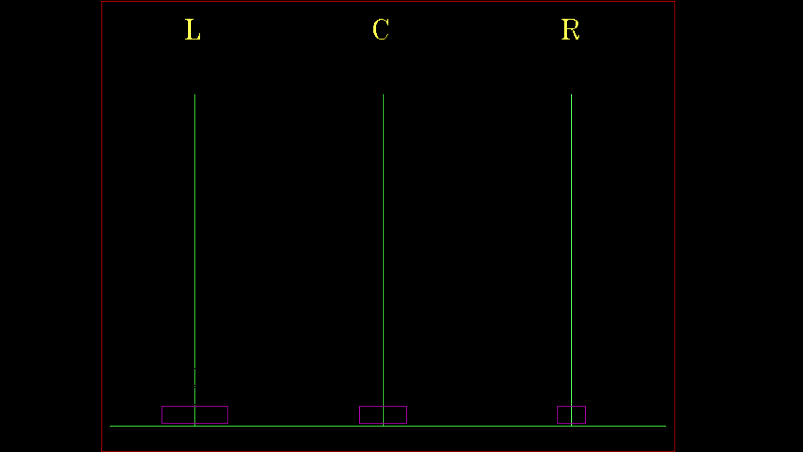
Initial position

****

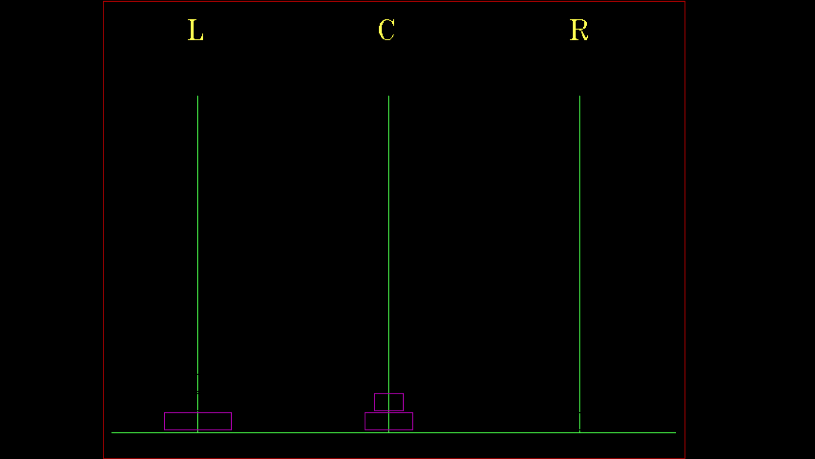
**1st Move**

****

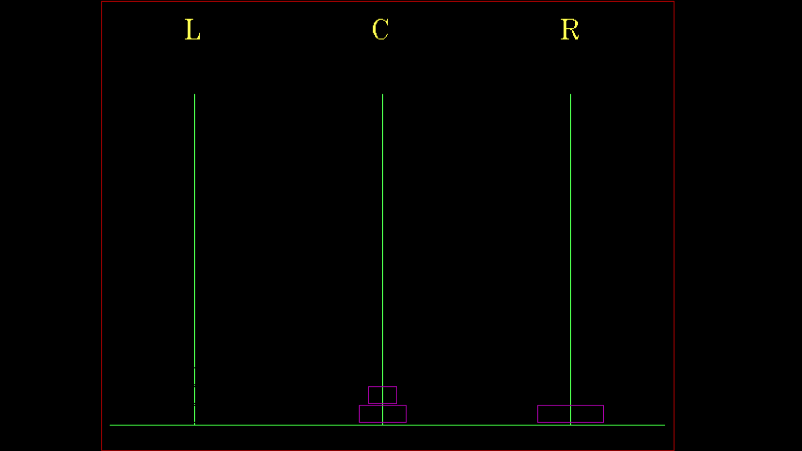
**2nd Move**

****

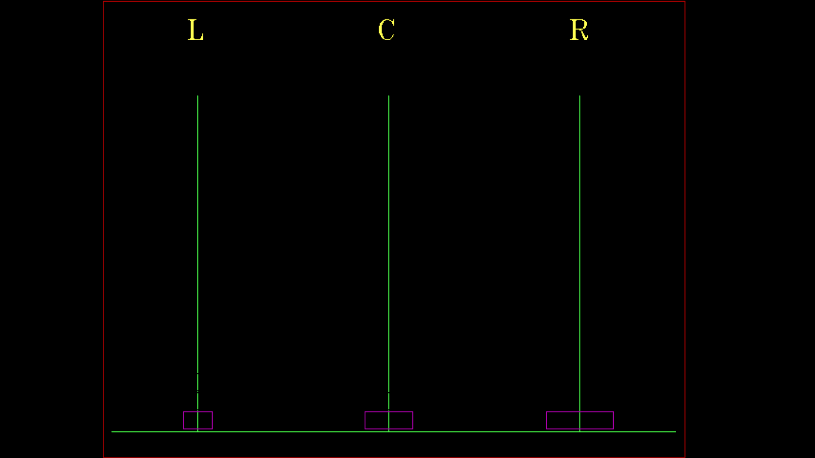
**3rd Move**

****

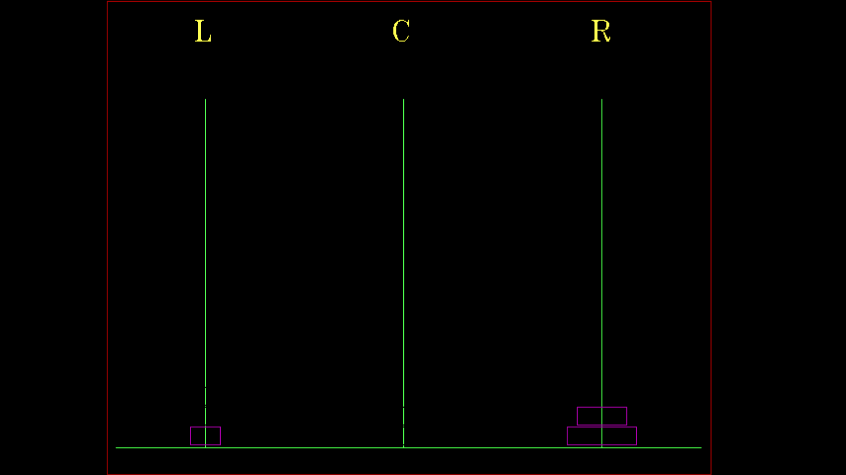
**4th Move**

****

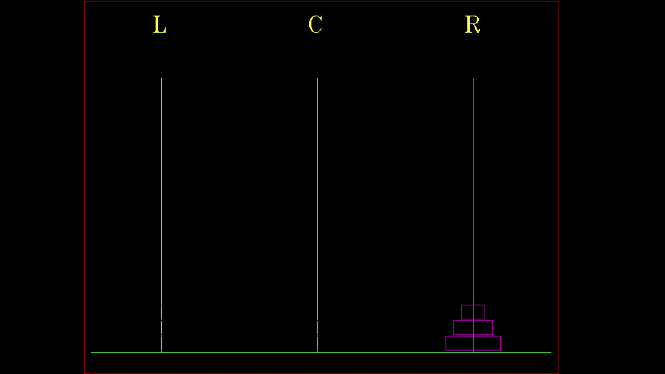
**5th Move**

****

**6th Move**

****

**7th Move/Final Position**

****

**CONCLUSION**

Thus, we come to the conclusion that for ‘n’ disks we need to make ‘(2^n)-1’ moves and so on.

**REFERENCE**

* en.wikipedia.org
* www.geeksforgeeks.org