DATA STRUCTURES AND ALGORITHMS

LAB ASSIGNMENT-5

NAME – KAPAROTU VENKATA SURYA THARANI

COURSE – AIDE

SECTION - “A”

USN ID – 22BTRAD018

Using recursive and non-recursive methods implement Tower of Hanoi.

# Recursive program to implement Tower of Hanoi:

**Code:**

**package** p1;

**public class** TowerOfHanoi {

**static void** towerOfHanoi(**int** x, **char** from\_rod, **char** to\_rod, **char**

helper\_rod)

{

**if** (x == 1)

{

System.***out***.println("Take disk 1 from rod " + from\_rod + " to rod " + to\_rod);

## return;

}

*towerOfHanoi*(x-1, from\_rod, helper\_rod, to\_rod); System.***out***.println("Take disk " + x + " from rod " + from\_rod + "

to rod " + to\_rod);

*towerOfHanoi*(x-1, helper\_rod, to\_rod, from\_rod);

}

**public static void** main(String args[])

{

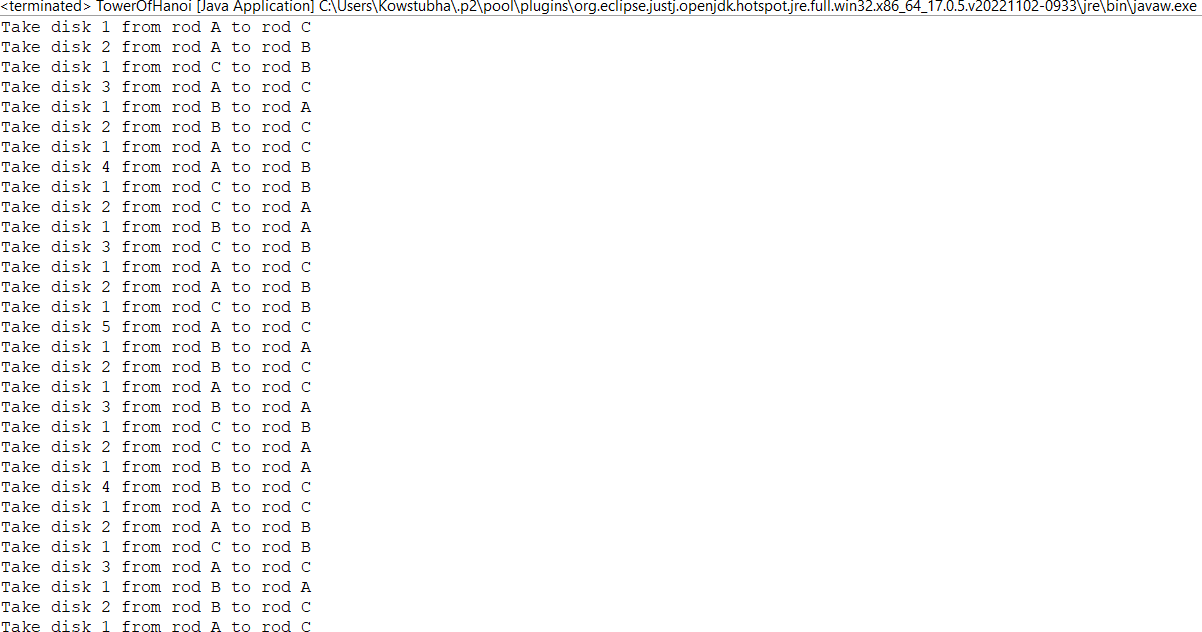
**int** n = 5;

*towerOfHanoi*(n,'A','C', 'B');

}

}

# Output:



**Non-recursive program to implement Tower of Hanoi:**

# Code:

**package** p1;

**public class** Tower\_of\_Hanoi {

**class** Stack

{

**int** capacity;

**int** top;

**int** arr[];

}

Stack createStack(**int** capacity)

{

Stack st1 = **new** Stack(); st1.capacity = capacity; st1.top = -1;

st1.arr = **new int**[capacity]; **return** st1;

}

**boolean** isFull(Stack st1)

{

**return** (st1.top == st1.capacity - 1);

}

**boolean** isEmpty(Stack st1)

{

**return** (st1.top == -1);

}

**void** push(Stack st1, **int** item)

{

**if** (isFull(st1))

## return;

st1.arr[++st1.top] = item;

}

**int** pop(Stack st1)

{

**if** (isEmpty(st1))

**return** Integer.***MIN\_VALUE***;

**return** st1.arr[st1.top--];

}

**void** moveDisksBetweenTwoPoles(Stack src, Stack dest,

**char** s, **char** d)

{

**int** p1TopDisk = pop(src);

**int** p2TopDisk = pop(dest);

**if** (p1TopDisk == Integer.***MIN\_VALUE***)

{

push(src, p2TopDisk); moveDisk(d, s, p2TopDisk);

}

**else if** (p2TopDisk == Integer.***MIN\_VALUE***)

{

push(dest, p1TopDisk); moveDisk(s, d, p1TopDisk);

}

**else if** (p1TopDisk > p2TopDisk)

{

push(src, p1TopDisk); push(src, p2TopDisk); moveDisk(d, s, p2TopDisk);

}

## else

{

push(dest, p2TopDisk); push(dest, p1TopDisk); moveDisk(s, d, p1TopDisk);

}

}

**void** moveDisk(**char** fromrod, **char** torod, **int** disk)

{

System.***out***.println("Move the disk " + disk +

" from " + fromrod + " to " + torod);

}

**void** toh\_nr(**int** no\_of\_disks, Stack

src, Stack aux, Stack dest)

{

**int** i, total\_no\_of\_moves;

**char** s = 'S', d = 'D', a = 'A';

**if** (no\_of\_disks % 2 == 0)

{

**char** temp = d; d = a;

a = temp;

}

total\_no\_of\_moves = (**int**)(Math.*pow*(

2, no\_of\_disks) - 1);

**for**(i = no\_of\_disks; i >= 1; i--) push(src, i);

**for**(i = 1; i <= total\_no\_of\_moves; i++)

{

**if** (i % 3 == 1) moveDisksBetweenTwoPoles(src, dest, s, d);

**else if** (i % 3 == 2) moveDisksBetweenTwoPoles(src, aux, s, a);

**else if** (i % 3 == 0) moveDisksBetweenTwoPoles(aux, dest, a, d);

}

}

**public static void** main(String[] args)

{

**int** no\_of\_disks = 5;

Tower\_of\_Hanoi toh1 = **new** Tower\_of\_Hanoi(); Stack src, dest, aux;

src = toh1.createStack(no\_of\_disks); dest = toh1.createStack(no\_of\_disks); aux = toh1.createStack(no\_of\_disks);

toh1.toh\_nr(no\_of\_disks, src, aux, dest);

}

}

# Output:

