Lab 18
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    vector<int> t[3];  // array of 3 vectors that stores int
    // three towers A,B,C represented as an array of 3 vectors
    int n, candidate, to, from, move = 0;
    // move counts the move number
    cout << "Please enter an ODD number of rings to move: ";
    cin >> n;
    /*We have one virtue large ring at the bottom of each tower for the ease of
    comparison. fill the tower with rings, tower A has all the disc, t[0].push_back(1)
    function add 1 to the back of tower A*/
    for (int i = n + 1; i >= 1; i--)
    { t[0].push_back(i);
    t[1].push_back(n + 1);
    t[2].push_back(n + 1);
    }
Towers of Hanoi Odd Rings

/* initialize towers and candidate for odd rings, the first move is to move the top ring from the source tower to the destination tower */
from=0;
to=1;
candidate=1;
while( t[1].size() !=n+1) {
    /* As long as the destination tower doesn’t have all the rings size() is an instance function that returns how many rings in the vector*/
Towers of Hanoi Odd Rings

//mimic the move in vectors
1. Push the ring to the “to” tower
2. Remove the ring from the “from” tower

candidate=t[from].back();
t[from].pop_back();
t[to].push_back(candidate);
Towers of Hanoi Odd Rings

//Setup for the next move, 
//look for the ring to move 
//we're not going to move the ring that we just moved 
//we're about to move the smaller ring of the other two towers 
// other towers → 
//(to + 1) % 3 index of the right tower 
//(to + 2) % 3 index of the left tower 
//check the last(top) ring of each tower 
if (t[(to + 1) % 3].back() < t[(to + 2) % 3].back())
    from =;
else
    from =;
//set the index to the tower with the smaller ring 
//get the ring to move from the tower 
//* Identify where you're going to be moving the next candidate */
if ( t[(from)%3].back()<(from+1)%3].back() )
    to = ; //move right
else
    to = ; //move left 
//This part of move will be different for even n