```cpp
#include <fstream>
#include <iostream>
#include "stack.cpp"
#include <unistd.h>
using namespace std;

struct Node {
    Stack * v;
    Node * next;
    Node * dest;
    Node(Stack * i) : v(i) {}
};

std::ofstream out("./outputs/test.txt");

class Graph {
private:
    Node *s, *a1, *a2, *a3, *d;
    int n, recursiveCalls, moves;
    bool animation, movesValid;
    double delay;
public:
    Graph(int n, bool animation = true, double delay = 0) :
        n(n), animation(animation), delay(delay), recursiveCalls(0),
        moves(0), movesValid(true){

        s = new Node(new Stack(n, "Start"));
        a1 = new Node(new Stack(n, "Aux1"));
        a2 = new Node(new Stack(n, "Aux2"));
        a3 = new Node(new Stack(n, "Aux3"));
        d = new Node(new Stack(n, "Dest"));

        s->next = a1;
        a1->next = a2;
        a2->next = a3;
        a3->next = a1;
        a3->dest = d;

        for(int i = n; i > 0;i--)
            s->v->push(i);

        Hanoi(n);
        out.close();
    }

    ~Graph() {
        delete s;
        delete a1;
        delete a2;
        delete a3;
        delete d;
    }

    void Hanoi(int n) {
        if(!s->v->isEmpty() && n >= 1) {
            Hanoi(n-1);
            moveNext(s, a2);
            H1(n-1, a3, a2, a1);
            moveNext(a2,a3);
            if(!s->v->isEmpty())
                H2(n-1, a1, a2, a3);
            else
                Hanoi(n-1);
        } else if(s->v->isEmpty()) {
            if(n == 0)
```

```
66                  moveNext(a3,d);
67              else if(n >= 1) {
68                  moveNext(a3, d);
69                  H2(n-1, a1, a2, a3);
70                  moveNext(a1, a2);
71                  H1(n-1, a3, a2, a1);
72                  moveNext(a2, a3);
73                  Hanoi(n-1);
74              }
75          } recursiveCalls++;
76      }
77
78
79      void H2(int n, Node * begin, Node * aux, Node * end) {
80          if(n == 1)
81              moveNext(begin, end);
82          else if(n >= 2) {
83              H2(n-1, begin, aux, end);
84              moveNext(begin, aux);
85              H1(n-1, end, aux, begin);
86              moveNext(aux, end);
87              H2(n-1, begin, aux, end);
88          } recursiveCalls++;
89      }
90
91      void H1(int n, Node * begin, Node * aux, Node * end) {
92          if(n == 1)
93              moveNext(begin, end);
94          else if(n >= 2) {
95              H2(n-1, begin, end, aux);
96              moveNext(begin, end);
97              H2(n-1, aux, begin, end);
98          } recursiveCalls++;
99      }
100
101     void moveNext(Node * a, Node * b) {
102         if(b == d) {
103             if(b == d && a == a3) {
104                 out << a->v->getName() << " -> " << a->v->top()
105                     << " -> " << d->v->getName() << std::endl;
106                 b->v->push(a->v->pop());
107             } else {
108                 out << a->v->getName() << " -> " << a->v->top()
109                     << " -> " << a->next->v->getName() << std::endl;
110                 a->next->v->push(a->v->pop());
111                 moveNext(a->next, b);
112             }
113         } else if(a->next == b) {
114             out << a->v->getName() << " -> " << a->v->top()
115                 << " -> " << b->v->getName() << std::endl;
116             b->v->push(a->v->pop());
117         } else {
118             out << a->v->getName() << " -> " << a->v->top()
119                 << " -> " << a->next->v->getName() << std::endl;
120             a->next->v->push(a->v->pop());
121             moveNext(a->next, b);
122         } moves++;
123     }
124 };
```