```
/* FILE: hanoi_recognize_conf.cpp   last change: 31-Aug-2012   author: Romeo Rizzi
 * a solver for problem "hanoi_recognize_conf" in the 28-09-2012 exam in Algorithms
 */

#define NDEBUG // NDEBUG disabilita tutte le assert se definita prima di includere <cassert>
#include <cassert>

#ifndef NDEBUG
#  include <iostream>  // non voglio includerla quando Non compilo in modalita' DEBUG
#endif
#include <fstream>

using namespace std;

const long long int UNKNOWN = -1;
const int MAX_N = 100000;
int n;   // number of disks: the disks are numbered from 1 (the smallest) to n (the biggest)
int pole_of_disk[MAX_N +1]; // the position of each disk in the current input question
long long int memo_num_mosse[MAX_N +1]; // offers memoization to procedure <num_mosse> here below

long long int num_mosse(int n) {
// minimo numero di mosse per spostare un'intera torre di n dischi da un piolo ad un altro
  if( memo_num_mosse[n] == UNKNOWN )  memo_num_mosse[n] = 1+2*num_mosse(n-1);
  return memo_num_mosse[n];
}

bool is_on_path_from_tower_to_tower(int n, int pole_from, int pole_to, int pole_aux) {
// considerate solo i dischi 1...<n> (i dischi piu' grandi non vanno mossi, fate come non ci fossero)
// ritorna true se la configurazione in input (come memorizzata nel vettore globale pole_of_disk)
// e' una delle configurazioni che appaiono sul piu' breve cammino
// dalla configurazione con l'intera torre di <n> dischi posta su <pole_from>
// alla configurazione con l'intera torre di <n> dischi posta su <pole_to>

  if(n <= 0) return true;
  if( pole_of_disk[n] == pole_from )  return is_on_path_from_tower_to_tower(n-1, pole_from, pole_aux, pole_to);
  if( pole_of_disk[n] == pole_to )
    // hai gia' spostato il disco grande e per farlo devi avere messo tutta la torre(n-1) sul pole_aux
    // a quel punto volevi portare la torre dal pole_aux al pole_to
    return is_on_path_from_tower_to_tower(n-1, pole_aux, pole_to, pole_from);

  return false;  // poiche' nello spostare un'intera torre il disco grande non va mai sul pole_aux
}

long long int num_mosse_on_path_from_tower_to_tower(int n, int pole_from, int pole_to, int pole_aux) {
// considerate solo i dischi 1...<n> (i dischi piu' grandi non vanno mossi, fate come non ci fossero)
// assume che  <is_on_path_from_tower_to_tower(n, pole_from, pole_to, pole_aux) == true>
// ritorna il numero di mosse che portano alla configurazione memorizzata nel vettore globale pole_of_disk
// quando ci si muova dalla configurazione con l'intera torre di <n> dischi posta su <pole_from>
// alla configurazione con l'intera torre di <n> dischi posta su <pole_to>
```



```
  //cout << "n = " << n << ", pole_from = " << pole_from << ", pole_to = "  << pole_to << ", pole_aux = "  << pole_aux << endl;
  if(n <= 0) return 0;
  if( pole_of_disk[n] == pole_from )  return num_mosse_on_path_from_tower_to_tower(n-1, pole_from, pole_aux, pole_to);
  assert( pole_of_disk[n] == pole_to );  // poiche' nello spostare un'intera torre il disco grande non va mai sul pole_aux
    // quindi hai gia' spostato il disco grande e per farlo devi avere messo tutta la torre(n-1) sul pole_aux
    // a questo punto dovrai ancora portare la torre(n-1) dal pole_aux al pole_to
    return num_mosse(n-1) + 1 + num_mosse_on_path_from_tower_to_tower(n-1, pole_aux, pole_to, pole_from);
}


int main() {
  ifstream fin("input.txt"); assert(fin);
  ofstream fout("output.txt"); assert(fout);
  fin >> n;
  memo_num_mosse[0] = 0; for(int i = 1; i <= n; i++)  memo_num_mosse[i] = UNKNOWN;
  for(int question = 1; question <= 3; question++) {
    for(int i = 1; i <= n; i++)  fin >> pole_of_disk[i];
    int pole_from = pole_of_disk[n];
    int biggest_disk_moved = 0; int pole_to = pole_from;
    for(int i = 1; i < n; i++)  if( pole_of_disk[i] != pole_from ) { pole_to = pole_of_disk[i]; biggest_disk_moved = i; }
    if( pole_from == pole_to) fout << 0 << endl;
    else {
      int pole_aux = 6 -pole_from -pole_to;
      if( is_on_path_from_tower_to_tower(biggest_disk_moved, pole_from, pole_to, pole_aux) )
        fout << num_mosse_on_path_from_tower_to_tower(biggest_disk_moved, pole_from, pole_to, pole_aux) << endl;
      else fout << -1 << endl;
    }
  }
  fin.close(); fout.close();
  return 0;
}
```