# biglie.cpp

```
/* FILE: biglie.cpp   last change: 27-Feb-2014   author: Romeo Rizzi
 * an O(2^n) time and memory solution for the game with marbles.
 */

//#define NDEBUG   // NDEBUG definita nella versione che consegno
#include <cassert>
#ifndef NDEBUG
#  include <iostream>  // uso di cin e cout non consentito in versione finale
#endif
#include <fstream>
#include <algorithm>

using namespace std;

const int UNKNOWN = -1;
const int N = 5;
const int N_CELLS = N*N;
const int POW_2_N_CELLS = 1024*1024*32;
int maxNumMoves[ POW_2_N_CELLS ];
int best_iA[ POW_2_N_CELLS ];
int best_jA[ POW_2_N_CELLS ];
int best_iB[ POW_2_N_CELLS ];
int best_jB[ POW_2_N_CELLS ];

int mymax( int a, int b ) { return (a>b)? a : b; }

long int code( int board[N][N] ) {
  long int risp = 0; int B = 1;
  for(int i = 0; i < N; i++)
    for(int j = 0; j < N; j++) {
      if( board[i][j] )
        risp += B;
      B *= 2;
    }
  return risp;
}

int maxMoves( int board[N][N] ) {
  long int codeBoard = code(board);
  if( maxNumMoves[codeBoard] != UNKNOWN )  return maxNumMoves[codeBoard];
  maxNumMoves[codeBoard] = 0;
  for(int iB = 1; iB <= N-2; iB++) for(int jB = 1; jB <= N-2; jB++)
    if( board[iB][jB] )
      for(int iA = iB-1; iA <= iB+1; iA++) for(int jA = jB-1; jA <= jB+1; jA++)
        if( (iA != iB) || (jA != jB) )  if( board[iA][jA] ) {
            int iC = 2*iB -iA;
            int jC = 2*jB -jA;
            if( board[iC][jC] == 0 ) {
               board[iA][jA] = 0; board[iB][jB] = 0; board[iC][jC] = 1;
               //long int codeBoardAfterMove = code(board); Commentando e spostando sotto verifico
che in effetti e' collo di bottiglia.
               //Quindi si potrebbe accellerare producendo il code per differenza da configurazione
 board precedente. Ma ho preferito assegnarvi un tempo largo.
               if( maxNumMoves[codeBoard] <= maxMoves(board) ) {
                  long int codeBoardAfterMove = code(board);
                  maxNumMoves[codeBoard] = maxNumMoves[codeBoardAfterMove] +1;
                  best_iA[ codeBoard ] = iA; best_jA[ codeBoard ] = jA;
                  best_iB[ codeBoard ] = iB; best_jB[ codeBoard ] = jB;
               }
                 board[iA][jA] = 1; board[iB][jB] = 1; board[iC][jC] = 0;
            }
        }
  return maxNumMoves[codeBoard];
}


int main() {
  int board[N][N];
  ifstream fin("input.txt"); assert( fin );
  for(int i = 0; i < N; i++)
     for(int j = 0; j < N; j++)
        fin >> board[i][j];
  fin.close();
  for(long int c = 0; c < POW_2_N_CELLS; c++)
        maxNumMoves[ c ] = UNKNOWN;
  int nMoves = maxMoves( board );
  ofstream fout("output.txt"); assert( fout );
```

**biglie.cpp**

```
  fout << nMoves << endl;
  long int codeBoard = code( board );
  for(int i = 0; i < nMoves; i++) {
    int iA = best_iA[ codeBoard ];  int jA = best_jA[ codeBoard ];
    int iB = best_iB[ codeBoard ];  int jB = best_jB[ codeBoard ];
    int iC = 2*iB −iA;
    int jC = 2*jB −jA;
    fout << iA+1 << " " << jA+1 << " "
         << iB+1 << " " << jB+1 << " "
         << iC+1 << " " << jC+1 << endl;
    board[iA][jA] = 0; board[iB][jB] = 0; board[iC][jC] = 1;
    codeBoard = code( board );
  }
  fout.close();
  return 0;
}
```