```
/* FILE: tournament.sol2.cpp   last change: 8-Sept-2012   author: Romeo Rizzi
 * a solver for problem 1 (tournament) in the 25-06-2012 exam in Algorithms
 *
 * Questa seconda soluzione muove da un'approccio piu' sfacciatamente induttivo (stesso Teo, nuova proof) e si rivela piu' essenziale.
 * Teorema: ogni tournament ammette un cammino Hamiltoniamo.
 * proof: Dato un tournament T di n nodi numerati 1,2, ..., n, chiediamo alla fatina ricorsina di consegnarci un cammino hamiltoniano H nel sottotournament ottenuto ignorando il nodo n.
 * Assumeremo che H sia memorizzato come la sequenza dei suoi nodi H = H[1], H[2], ..., H[n-1].
 * Se H[n-1] batte n, allora ritorniamo la sequenza H' = H, n.
 * Se nessuno di questi due facili e fortuiti casi si applica, allora esiste un i tale che:
 *   (H[i] batte n) .AND. (n batte H[i+1])
 * e ce la caviamo inserendo n proprio tra H[i] ed H[i+1].  QED
 */

// #include <iostream>  // NOTA: ho commentato questa riga per essere sicuro di non aver lasciato operazioni di input/output di debug nella fretta di consegnare, se compila cosi' non ci sono ed io non perdo stupidamente i miei punti-esame.
#include <fstream>
#include <cassert>

using namespace std;

const int MAX_N = 2000; int n;   // Numero giocatori (numero di nodi nel tournament, un tournament e' un grafo diretto completo).

// Matrice vittorie/sconfitte: T[i][j] = ture se i batte j
bool T[MAX_N +1][MAX_N +1];   // i nodi sono numerati da 1 ad n.

int H[MAX_N +1]; // il cammino hamiltoniano H dato come sequenza di n nodi.

int main() {
  ifstream fin("input.txt");  assert( fin );
  fin >> n;
  for(int i = 1; i <= n; i++)
    for(int j = 1; j <= n; j++)
      fin >> T[i][j];
  fin.close();

  H[1] = 1;
  for(int new_node = 2; new_node <= n; new_node++) {
    int insert_pos = new_node;
    while( (insert_pos >= 2) & T[new_node][ H[insert_pos-1] ] ) {
     H[insert_pos] = H[insert_pos-1];
      insert_pos--;
    }
    H[insert_pos] = new_node;
  }

  ofstream fout("output.txt");  assert( fout );
  fout << n << endl;
  for(int i = 1; i < n; i++)  fout << H[i] << " " << H[i+1] << endl;
  fout.close();

  return 0;
}
```