permutGame.cpp

```
/* FILE: permutGame.cpp   last change: 2-Jul-2013   author: Romeo Rizzi
 * a solver for problem permutGame
 */

//#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>

using namespace std;

const int MAX_N = 10000;
const int MAX_K = 100;
int n, k;

int p[MAX_K][MAX_N+1]; // le k permutazioni di 1,2, ..., n che definiscono la configurazione del gi
oco

bool got_sorted[MAX_K];

int swap1[MAX_N+2][MAX_K], swap2[MAX_N+2][MAX_K];
// alla mossa t, nella permutazione p[j], vengono scambiati i due elementi nelle posizioni swap1[t]
[j], swap2[t][j]
/* nota: le mosse sono massimo MAX_N
   su ognuna delle k permutazioni,
   noi scambiamo il primo elemento fuori posto con l'elemento che e' al suo posto, fino a quando la
 permutazione e' ordinata. Da qui in poi oscilliamo i primi due elementi
   Terminiamo quando ciascuna permutazione e' stata ordinata almeno una volta ed almeno meta' delle
 permutazioni sono attualmente ordinate.
   Infatti la parita' e' l'unica invariante che limita la nostra capacita' di ordinare.
*/

void swap( int &a, int &b ) { int tmp = a; a = b; b = tmp; }

int main() {
  ifstream fin("input.txt"); assert( fin );
  fin >> n >> k;
  for(int j = 0; j < k; j++) {
     got_sorted[j] = false;
     for(int i = 1; i <= n; i++)
       fin >> p[j][i];
  }
  fin.close();

  int num_sorted, num_got_sorted = 0, num_mosse = 0;
  bool finished = false;
  while( !finished ) {
    num_mosse++;
    num_sorted = 0;
    for(int j = 0; j < k; j++) {
      swap1[num_mosse][j] = 1;
      if( got_sorted[j] ) swap2[num_mosse][j] = 2;
      else {
        while( (swap1[num_mosse][j] <= n) && ( p[j][swap1[num_mosse][j]] == swap1[num_mosse][j] ) )
 swap1[num_mosse][j]++;
        if ( swap1[num_mosse][j] > n ) {
           got_sorted[j] = true; num_got_sorted++;
           swap1[num_mosse][j] = 1; swap2[num_mosse][j] = 2;
        }
        else swap2[num_mosse][j] = p[j][swap1[num_mosse][j]];
      }
      swap(p[j][swap1[num_mosse][j]], p[j][swap2[num_mosse][j]]);
      if( got_sorted[j] && (p[j][1] == 1) ) num_sorted++;
    }
    if( ( num_got_sorted == k ) && ( num_sorted >= (k+1)/2 ) )  finished = true;
  }

  ofstream fout("output.txt"); assert( fout );
  fout << num_sorted << " " << num_mosse << endl;
  for(int t = 1; t <= num_mosse; t++) {
     for(int j = 0; j < k; j++)
       fout << swap1[t][j] << " " << swap2[t][j] << " ";
     fout << endl;
  }
  fout.close();
```

permutGame.cpp

```
  return 0;
}
```