EulerTree.cpp

```
/* FILE: EulerTree.cpp   last change: 26-Jan-2013   author: Romeo Rizzi
 * a solver for problem EulerTree
 */

#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 = 100;  // massima numero di nodi;
int n, m; // numero di nodi e numero di archi

bool seen[MAX_N]; int degree[MAX_N];
int LIFOstack[MAX_N], LIFOpos = 0;

int main() {
  ifstream fin("input.txt"); assert( fin );
  fin >> n >> m;
  for(int i = 0; i < n; i++) { seen[i] = false; degree[i] = 0; }
  int cost = (n-1) + m*(m-n+1); // all edges are traversed at least once. The n-1 tree edges cost 1
 and the other cost m;
  int a, b, v = 0; seen[v] = true;
  for(int i = 1; i <= m; i++) {
    fin >> a >> b;  degree[a]++; degree[b]++;
    while( a != v ) { // time to backtrack through tree edges
      if( degree[v]%2 ) { // the tree edge which has lead to v and we are now traversing back must
be doubled to even out the parity of the degree of v.
         cost++;
         v = LIFOstack[ --LIFOpos ];
         degree[v]++;
      }
      else v = LIFOstack[ --LIFOpos ]; // we backtrack without doubling the edge
    }
    if( !seen[b] ) {
      seen[b] = true;
      LIFOstack[ LIFOpos++ ] = v;
      v = b;
    }
  }
  while( 0 != v ) { // time to backtrack through tree edges
    if( degree[v]%2 ) { // the tree edge which has lead to v and we are now traversing back must be
 doubled to even out the parity of the degree of v.
       cost++;
       v = LIFOstack[ --LIFOpos ];
       degree[v]++;
    }
    else v = LIFOstack[ --LIFOpos ]; // we backtrack without doubling the edge
  }
  fin.close();
  ofstream fout("output.txt"); assert( fout );
  fout << cost << endl;
  fout.close();
  return 0;
}
```