```
/* FILE: solver.cpp   last change: 10-Feb-2015   author: Romeo Rizzi
 * a linear solver for problem "match2n"
 */

#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 = 1000000;
int n; int S[2][MAX_N]; // la strip in input di dimensioni 2xn.

int opt[MAX_N+1];   // opt[i] = the minimum cost of a matching for the substrip made of the fi
rst i columns of S
int num[MAX_N+1];   // num[i] = the number (mod 1000000) of optimal matchings for the substrip
 made of the first i columns of S

int dist(int a, int b) { return ( a-b > 0 ) ? a-b : b-a; }

int main() {
  ifstream fin("input.txt"); assert( fin );
  fin >> n;
  for(int i = 0; i < n; i++)  fin >> S[0][i];
  for(int i = 0; i < n; i++)  fin >> S[1][i];
  fin.close();

  opt[0] = 0; num[0] = 1;
  opt[1] = dist( S[0][0], S[1][0] ); num[1] = 1;

  for(int i = 2; i <= n; i++) {
      opt[i] = opt[i-1] + dist( S[0][i-1], S[1][i-1] );
      num[i] = num[i-1];
      int second_case_cost = opt[i-2] + dist( S[0][i-1], S[0][i-2] )
                                      + dist( S[1][i-1], S[1][i-2] );
      if( opt[i] == second_case_cost ) {
        num[i] = ( num[i-1] + num[i-2] ) % 1000000;
      }
      if( opt[i] > second_case_cost ) {
         opt[i] = second_case_cost;
         num[i] = num[i-2];
      }
  }

  ofstream fout("output.txt");
  fout << opt[n] << endl << num[n] << endl;
  fout.close();
  return 0;
}
```