parenthesesMaskSimple.cpp

```
/* FILE: parenthesesMaskSimple.cpp   last change: 30-Jul-2013   author: Romeo Rizzi
 * a first scratch solver for problem parenthesesMask based on dynamic programming
 * this version can not deal with very big inputs since it does not exploit the fact that the asteri
sks are less than the length of the string;
 * the basic DP ideas should however appear more transparent and immidiate for a more readable code.
 Developed for comparison and scoring issues. (And also as first prototype).
 */

#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 BASE = 100000;
const int MAX_N = 10000;  // massima lunghezza della stringa in input
char p[MAX_N+1]; // la stringa pattern data in input
int n;  // lunghezza della stringa pattern in input

int opt[MAX_N +1][MAX_N +1];
/* opt[i][j] = the number of strings s over the alphabet {'(',')'} such that:
1. s is a prefix of a well formed parenthesis formula;
2. s matches the prefix p[1,..., i] of the pattern;
3. in s the number of '(' exceeds by precisely j the number of ')'.
Thus opt[n][0] is the answer to our problem.
The values actually stored in opt[][] are correct only modulo BASE.
*/

int main() {
  ifstream fin("input.txt"); assert( fin );
  fin >> n;
  for(int i = 1; i <= n; i++)  fin >> p[i];
  fin.close();
  opt[0][0] = 1;
  for(int j = 1; j <= n; j++)  opt[0][j] = 0;
  for(int i = 1; i <= n; i++)
    for(int j = 0; j <= n; j++)
      if( (i + j) % 2 ) opt[i][j] = 0;
      else switch( p[i] ) {
           case ')': opt[i][j] = opt[i-1][j+1]; break;
           case '(': opt[i][j] = ( j > 0 ) ? opt[i-1][j-1] : 0; break;
           case '*': opt[i][j] = opt[i-1][j+1]; if( j > 0 ) opt[i][j] += opt[i-1][j-1] % BASE; break
;
        default: assert( false );
      }

  //for(int i = 0; i <= n; i++) {
  //  for(int j = 0; j <= n; j++)  cout << opt[i][j] << " ";
  //  cout << endl;
  //}

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