E - PolandBall and Hypothesis

Código

```
#include <iostream>
#include <string>

//
// https://www.geeksforgeeks.org/prime-numbers/
//
bool is_prime(int n, int i)
{
  if( n == 0 || n == 1 )
    return false;

  if( n == i )
    return true;

  if( n % i == 0)
    return false;

  i += 1;

  return is_prime(n,i);

}

int main() {
    int n;
    std::cin >> n;

    for( int m = 1; m < 1001; m++)
    {
      int p = n*m + 1;
      if( ! is_prime( p, 2 ) )
      {
          std::cout << m << std::endl;
          break;
      }
    }
    return 0;
}
```

Corretude

O algoritmo funciona porque ele varre todos os valores de **m** dentro do domínio especificado pelo problema de [1, 1000] de modo que todos os valores **n*m + 1**
passam por um verificação de ser primo. Inevitavelmente algum valor de **n*m + 1** será não primo. Essa valor é capturado pela cláusula **If** que dentro do seu escopo conduz para a impressão do número **m** e finalização do algoritmo.

F - Dreamoon and WiFi

Código

```
def main():
    stg1 = ''.join( sorted( input() ) )
    stg2 = ''.join( sorted( input() ) )

    bag = ['']

    for s in stg2:
        if s == '?':
            tmp = []
            for stg in bag:
                tmp.append( '+' + stg )
                tmp.append( stg + '-' )
            bag = tmp

        if s == '+':
            for en, stg in enumerate(bag):
                bag[en] = '+' + stg

        if s == '-':
            for en, stg in enumerate(bag):
                bag[en] = stg + '-'

    count = 0
    for stg in bag:
        if stg == stg1:
            count += 1

    print(count/ len(bag))

if __name__ == '__main__':
    main()
```

Corretude

Neste problema, o algoritmo funciona porque nele todo o espaço amostral das possíveis combinações de sinais ‘+’ e ‘-’ são computadas. Depois, são computadas todas as combinações de interesse e, por fim, o valor da probabilidade das combinações de interesse com relação ao espaço amostral de combinações é calculado. É importante mencionar que a única restrição do problema é que Dreamoon chegue no local final especificado pelas instruções vindas de Drazil corretamente. Dreamoon não precisa percorrer exatamente a sequência de instruções vindas de Drazil ou seja Dreamoon só precisa fazer os mesmo números de instruções ‘+’ e ‘-’ que as vindas de Drazil. Isso permitiu a montagem dos caminhos de modo que os sinais de ‘+’ sempre estão à esquerda dos sinais de ‘-’. Essa montagem conveniente, facilitou no processo de encontrar as combinaçẽos de sinais que conduzem para os mesmo local de chegada porque todos eles ficam iguais.