Binary diagnostic X85106_en

Given a '0'-'1' matrix where each row represents a binary number with its leftmost bit as the most significant one, we want to generate two new binary numbers called the *gamma* and *epsilon* rates, along with a new decimal number called *consumption*. These three numbers are generated as follows.
Each bit in the gamma rate can be determined by finding the most common bit in the corresponding position of all numbers in the matrix. For example, the following matrix:

```
00100
11110
10110
```

represents the binary numbers 00100, 11110 and 10110. Considering only the first bit of each number (the most significant one), there are two 1 bit and one 0 bit. Since the most common bit is 1, the first bit of the gamma rate is 1. The most common second bit of the numbers in the matrix is 0, so the second bit of the gamma rate is 0. The most common value of the third, fourth, and fifth bits are 1, 1, and 0, respectively, and so the final three bits of the gamma rate are 110. The gamma rate is the binary number 10110, or 22 in decimal. When in a given position the number of 0 bits and 1 bits is the same, we consider 0 as being the most common bit.
The epsilon rate is calculated in a similar way; rather than using the most common bit, the least common bit from each position is used. So, the epsilon rate is 01001, or 9 in decimal. When in a given position the number of 0 bits and 1 bits is the same, we consider 1 as being the least common bit.
Multiplying the gamma rate (22) by the epsilon rate (9) produces the power consumption, 198 in this example.
You MUST complete the following code:

```
typedef vector<char> Binario;
typedef vector<Binario> Matriz;

// Pre: n > 0, m > 0
// Post: retorna una matriz de '0' y '1' con dimensiones n*m leída de la entrad
Matriz leer_matriz(int n, int m) {
    ...
}

// Pre: -
// Post: escribe el string s, seguido por dos punts, seguido por el Binario b
void escribir(string s, const Binario& b) {
    cout << s << ":";
    for (int i = 0; i < b.size(); ++i) cout << " " << b[i];
    cout << endl;
}

// Pre: gamma.size() = epsilon.size()
// Post: retorna el consumo calculado como gamma*epsilon en decimal
int calcula_consumo(const Binario& gamma, const Binario& epsilon) {
    ...
}

// Pre: mat.size > 0; 0 <= j < mat[0].size()
// Post: retorna el bit más frecuente en la columna j de mat. En caso de empate
char bit_mas_comun(const Matriz& mat, int j) {
    ...
}

// Pre: mat.size > 0; gamma.size() == mat[0].size(); epsilon.size() == mat[0].s
// Post: gamma y epsilon son el ratio gamma y epsilon de mat, respectivamente
void calcula_ratios(const Matriz& mat, Binario& gamma, Binario& epsilon) {
    ...
}

int main() {
int n, m;
    while (cin >> n >> m) {
        Matriz mat = leer_matriz(n, m);
        Binario gamma(m);
        Binario epsilon(m);
        calcula_ratios(mat, gamma, epsilon);
        escribir("Gamma", gamma);
        escribir("Epsilon", epsilon);
        cout << "Consumo: " << calcula_consumo(gamma, epsilon) << endl << endl;
    }
}
```

Exam score: 3 **Automatic part:** 40%

Input

The input is a sequence of cases. Each case starts with two integers $n > 0$ and $m > 0$ followed by an $n \times m$ '0'-'1' matrix.

Output

For each case, the program writes the gamma and epsilon rates, along with the consumption represented by its matrix, followed by an empty line. See the examples.

Sample input

```
3 5
00100
11110
10110

4 5
00100
11110
10110
01001

1 1
1

2 1
0
1
```

```
1 15
111111111111110
```

Sample output

```
Gamma: 1 0 1 1 0
Epsilon: 0 1 0 0 1
Consumo: 198

Gamma: 0 0 1 0 0
Epsilon: 1 1 0 1 1
Consumo: 108

Gamma: 1
Epsilon: 0
Consumo: 0

Gamma: 0
Epsilon: 1
Consumo: 0

Gamma: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
Epsilon: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Consumo: 32766
```

Problem information

Author : Pro1
Generation : 2022-01-14 12:57:00