

```
1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4 #include <algorithm>
5 #include <utility>
6
7 using namespace std;
8
9 // DEFINICIONES DE TIPO
10 typedef pair<int, int> posicion;
11 typedef int altura;
12 typedef int poblacion;
13 typedef pair<altura, poblacion> zona;
14 typedef vector<vector<zona > > mapa;
15
16 vector<zona> generarVecinos(mapa m, posicion p) {
17     vector<zona> res;
18
19     if (p.first < m.size() - 1) {
20         res.push_back(m[p.first + 1][p.second]);
21     }
22
23     if (p.first > 0) {
24         res.push_back(m[p.first - 1][p.second]);
25     }
26
27     if (p.second < m[0].size() - 1) {
28         res.push_back(m[p.first][p.second + 1]);
29     }
30
31     if (p.second > 0) {
32         res.push_back(m[p.first][p.second - 1]);
33     }
34     return res;
35 }
36
37 bool esValle(mapa m, posicion p){
38     vector<zona> zonasVecinas = generarVecinos(m, p);
39     int cantVecinos = zonasVecinas.size();
40
41     int i = 0;
42     while (i < cantVecinos && zonasVecinas[i].first >= m[p.first][p.second].first) {
43         i++;
44     }
45     return i == cantVecinos;
46 }
47
48 int main() {
49     /* No hace falta modificar el main */
50
51     // La posición
52     int pos1;
53     cin >> pos1;
54     int pos2;
55     cin >> pos2;
56     posicion p = make_pair(pos1, pos2);
57
58     // El mapa
59     int f;
60     cin >> f;
61     int c;
62     cin >> c;
63     mapa m;
64     char aux;
65     int alt;
66     int pob;
67     for (int i = 0; i < f; i++) {
68         vector<zona> fila;
69         for (int j = 0; j < c; j++) {
70             cin >> aux;
71             cin >> alt;
72             cin >> pob;
```

```
73         cin >> aux;
74         fila.push_back(make_pair(aux, pob));
75     }
76
77     m.push_back(fila);
78 }
79
80 // Evaluo si la posición es valle
81 bool res = esValle(m, p);
82
83 // Imprimo la salida
84 cout << (res?"True":"False") << endl;
85
86 return 0;
87 }
88
```

```

1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4 #include <algorithm>
5 #include <utility>
6
7 using namespace std;
8
9 // DEFINICIONES DE TIPO
10 typedef pair<int, int> posicion;
11 typedef int altura;
12 typedef int poblacion;
13 typedef pair<altura, poblacion> zona;
14 typedef vector<vector<zona > > mapa;
15
16 void miPropioSwap(vector<posicion> &v, int i, int j) {
17     posicion tmp = v[i];
18     v[i] = v[j];
19     v[j] = tmp;
20 }
21
22 int poblacionDePos(mapa m, posicion p){
23     return m[p.first][p.second].second;
24 }
25
26 void ordenar(vector<posicion> &v, mapa m2) {
27     // Bubble Sort
28     for (int i = 0; i < v.size(); i++) {
29         for (int j = v.size() - 1; j > i; j--) {
30             if (poblacionDePos(m2, v[j]) < poblacionDePos(m2, v[j - 1])) {
31                 miPropioSwap(v, j, j - 1);
32             }
33         }
34     }
35 }
36
37
38 void reiniciarVector(vector<posicion> &v) {
39     int i = 0;
40     while (i < v.size()) {
41         v.pop_back();
42         i++;
43     }
44 }
45
46 int crecimientoPoblacional(mapa m1, mapa m2, posicion p) {
47     return m2[p.first][p.second].second - m1[p.first][p.second].second;
48 }
49
50 vector<posicion> bajaNatalidad(mapa m1, mapa m2) {
51     // Tomo como mínimo el crecimiento poblacional de la primera pos
52     int min = crecimientoPoblacional(m1, m2, make_pair(0,0));
53     vector<posicion> res;
54
55     for (int i = 0; i < m1.size(); i++) {
56         for (int j = 0; j < m1[0].size(); j++) {
57             posicion p = make_pair(i, j);
58             if (crecimientoPoblacional(m1, m2, p) == min) {
59                 res.push_back(p);
60             } else if (crecimientoPoblacional(m1, m2, p) < min) {
61                 reiniciarVector(res);
62                 min = crecimientoPoblacional(m1, m2, p);
63                 res.push_back(p);
64             }
65         }
66     }
67
68     ordenar(res, m2);
69
70     return res;
71 }
72

```

```
73 int main() {
74     /* No hace falta modificar el main */
75
76     // El mapa 1
77     int f;
78     cin >> f;
79     int c;
80     cin >> c;
81     mapa m;
82     char aux;
83     int alt;
84     int pob;
85     for (int i = 0; i < f; i++) {
86         vector<zona> fila;
87         for (int j = 0; j < c; j++) {
88             cin >> aux;
89             cin >> alt;
90             cin >> aux;
91             cin >> pob;
92             cin >> aux;
93             fila.push_back(make_pair(alt, pob));
94         }
95
96         m.push_back(fila);
97     }
98
99     // El mapa 2
100    int f2;
101    cin >> f2;
102    int c2;
103    cin >> c2;
104    mapa m2;
105    char aux2;
106    int alt2;
107    int pob2;
108    for (int i = 0; i < f2; i++) {
109        vector<zona> fila2;
110        for (int j = 0; j < c2; j++) {
111            cin >> aux2;
112            cin >> alt2;
113            cin >> aux2;
114            cin >> pob2;
115            cin >> aux2;
116            fila2.push_back(make_pair(alt2, pob2));
117        }
118
119        m2.push_back(fila2);
120    }
121
122    // Invoco la función
123    vector<posicion> pos = bajaNatalidad(m, m2);
124
125    // Ordeno el vector
126    // Deja de ser necesario porque ahora va ordenado de acuerdo a especificación std::sort(pos
    .begin(), pos.end());
127
128    // Imprimo la salida
129    for (int i = 0; i < pos.size(); i++) {
130        cout << "(" << pos[i].first << "," << pos[i].second << ")" << " ";
131    }
132
133    return 0;
134 }
135
```

```
1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4 #include <algorithm>
5 #include <utility>
6
7 using namespace std;
8
9 // DEFINICIONES DE TIPO
10 typedef pair<int, int> posicion;
11 typedef int altura;
12 typedef int poblacion;
13 typedef pair<altura, poblacion> zona;
14 typedef vector<vector<zona > > mapa;
15
16 void vivenEnAltura(mapa m, int &alt, int &cantGente) {
17     // Tomo como altura máxima a la altura de la primera pos
18     int max = m[0][0].first;
19     int sumaPoblacion = 0;
20
21     for (int i = 0; i < m.size(); i++) {
22         for (int j = 0; j < m[0].size(); j++) {
23             int alturaPos = m[i][j].first;
24             int poblacionPos = m[i][j].second;
25             if (alturaPos == max) {
26                 sumaPoblacion += poblacionPos;
27             } else if (alturaPos > max) {
28                 max = alturaPos;
29                 sumaPoblacion = poblacionPos;
30             }
31         }
32     }
33
34     alt = max;
35     cantGente = sumaPoblacion;
36 }
37
38 int main() {
39     /* No hace falta modificar el main */
40
41     // EL mapa 1
42     int f;
43     cin >> f;
44     int c;
45     cin >> c;
46     mapa m;
47     char aux;
48     int alt;
49     int pob;
50     for (int i = 0; i < f; i++) {
51         vector<zona> fila;
52         for (int j = 0; j < c; j++) {
53             cin >> aux;
54             cin >> alt;
55             cin >> aux;
56             cin >> pob;
57             cin >> aux;
58             fila.push_back(make_pair(alt, pob));
59         }
60
61         m.push_back(fila);
62     }
63
64     // Invoco la función
65     int altura;
66     int cantGente;
67     vivenEnAltura(m, altura, cantGente);
68
69     // Imprimo la salida
70     cout << altura << " " << cantGente << endl;
71
72     return 0;
}
```

73 }

74

```

1 #include <iostream>
2 #include <vector>
3 #include <cmath>
4 #include <algorithm>
5 #include <utility>
6
7 using namespace std;
8
9 // DEFINICIONES DE TIPO
10 typedef pair<int, int> posicion;
11 typedef int altura;
12 typedef int poblacion;
13 typedef pair<altura, poblacion> zona;
14 typedef vector<vector<zona > > mapa;
15
16 vector<zona> generarVecinos(mapa m, posicion p) {
17     vector<zona> res;
18
19     if (p.first < m.size() - 1) {
20         res.push_back(m[p.first + 1][p.second]);
21     }
22
23     if (p.first > 0) {
24         res.push_back(m[p.first - 1][p.second]);
25     }
26
27     if (p.second < m[0].size() - 1) {
28         res.push_back(m[p.first][p.second + 1]);
29     }
30
31     if (p.second > 0) {
32         res.push_back(m[p.first][p.second - 1]);
33     }
34     return res;
35 }
36 bool esValle(mapa m, posicion p){
37     vector<zona> zonasVecinas = generarVecinos(m, p);
38     int cantVecinos = zonasVecinas.size();
39     int alturaPosActual = m[p.first][p.second].first;
40
41     int i = 0;
42     while (i < cantVecinos && zonasVecinas[i].first >= alturaPosActual) {
43         i++;
44     }
45     return i == cantVecinos;
46 }
47
48 int promedioAlturasVecinas(mapa const &m, posicion const &p) {
49     vector<zona> vecinos = generarVecinos(m, p);
50     int cantVecinos = vecinos.size();
51     int sumaAlturasVecinas = 0;
52     for (int i = 0; i < cantVecinos; i++) {
53         sumaAlturasVecinas += vecinos[i].first;
54     }
55
56     return sumaAlturasVecinas / cantVecinos;
57 }
58
59 int rellenarValles(mapa &m) {
60     mapa m0 = m;
61     int sumaDifAlturas = 0;
62
63     for (int i = 0; i < m.size(); i++) {
64         for (int j = 0; j < m[0].size(); j++) {
65             posicion p = make_pair(i, j);
66             if (esValle(m0, p)) {
67                 m[i][j].first = promedioAlturasVecinas(m0, p);
68                 sumaDifAlturas += m[i][j].first - m0[i][j].first;
69             }
70         }
71     }
72

```

```
73     return sumaDifAlturas;
74 }
75
76 int main() {
77     /* No hace falta modificar el main */
78
79     // El mapa 1
80     int f;
81     cin >> f;
82     int c;
83     cin >> c;
84     mapa m;
85     char aux;
86     int alt;
87     int pob;
88     for (int i = 0; i < f; i++) {
89         vector<zona> fila;
90         for (int j = 0; j < c; j++) {
91             cin >> aux;
92             cin >> alt;
93             cin >> aux;
94             cin >> pob;
95             cin >> aux;
96             fila.push_back(make_pair(alt, pob));
97         }
98
99         m.push_back(fila);
100     }
101
102     // Invoco la función
103     int res = rellenarValles(m);
104
105     // Imprimo la salida
106     cout << res << endl;
107     cout << m.size() << endl;
108     cout << m[0].size() << endl;
109     for (int i = 0; i < m.size(); i++) {
110         for (int j = 0; j < m[0].size(); j++) {
111             cout << "(" << m[i][j].first << "," << m[i][j].second << ") ";
112         }
113         cout << endl;
114     }
115
116     return 0;
117 }
118
```