Turno 1

Best-Seller

```
#include <iostream>
#include <vector>
using namespace std;

struct Book {
    string title;       // titulo del libro
    int nb_pag;         // numero de paginas del libro
};

struct Author {
    string name;        // nombre del autor
    Book bestseller;    // libro mejor vendido del autor
};

typedef vector<Author> Library;

// Pre: en la entrada hay una secuencia que representa la info de los autores
//      tal y como describe el enunciado
// Post: devuelve una Library a partir de los datos de la entrada
Library read_data() {
    int n;
    cin >> n;
    Library lbr(n);
    for (int i = 0; i < n; ++i) {
        cin >> lbr[i].name >> lbr[i].bestseller.title >> lbr[i].bestseller.
           nb_pag;
    }
    return lbr;
}

int main() {
    Library lbr = read_data();
    char c;
    int p;
    while (cin >> c >> p) {
       cout << "-- begin query" << endl;
       for (int i = 0; i < lbr.size(); ++i) {
         if (lbr[i].bestseller.title[0] == c and lbr[i].bestseller.nb_pag > p)
             cout << lbr[i].name << endl;
       }
       cout << "-- end query" << endl;
```

```
    }
}
```

Posiciones de acumulación (1)

```
#include <iostream>
#include <vector>
using namespace std;

// Pre: en la entrada hay una secuencia de v.size() enteros
// Post: el vector v almacena los enteros de la entrada en el mismo
//           orden en el que aparecen
void leer_vector(vector<int>& v) {
    int n = v.size();
    for (int i = 0; i < n; ++i) cin >> v[i];
}

// Pre: los elementos de v son no negativos
// Post: retorna la cantidad de posiciones de acumulacin de v
int num_cumulos(const vector<int>& v) {
    int contador = 0;
    int n = v.size();
    for (int i = 0; i < n; ++i) {
        int suma = 0;
        int j = i - 1;
        while (j >= 0 and suma < v[i]) {
           suma += v[j];
          --j;
        }
       if (suma == v[i]) ++contador;
    }
   return contador;
}

int main() {
    int n;
    while (cin >> n) {
        vector<int> v(n);
        leer_vector(v);
        cout << num_cumulos(v) << endl;
    }
}
```

Número columnas diferentes

```
#include <iostream>
```

```cpp
#include <vector>
using namespace std;

typedef vector<int> Row;
typedef vector<Row> Matrix;

//reads a matrix from input
void read_mat(Matrix& mat) {
    int m = mat.size();
    int n = mat[0].size();
    for (int i = 0; i < m; ++i)
        for (int j = 0; j < n; ++j)
            cin >> mat[i][j];
}

//pre: i and j are two valid column indexes of mat
//post: returns true if columns i and j are the same
//      Returns false otherwise
bool are_equal(const Matrix& mat, int i, int j) {
    int m = mat.size();
    for (int k = 0; k < m; ++k)
        if (mat[k][i] != mat[k][j]) return false;
    return true;
}

//pre: i is valix index column of mat
//post: returns true when column i is different from  any previous one
//      Returns false otherwise
bool is_new_column(const Matrix& mat, int i) {
    for (int j = 0; j < i; ++j)
        if (are_equal(mat, j, i)) return false;
    return true;
}

int main() {
    int m, n;
    while (cin >> m >> n) {
        Matrix mat(m, Row(n));
        read_mat(mat);
        int counter = 0;
        for (int i = 0; i < n; ++i)
            if (is_new_column(mat,i)) ++counter;
        cout << counter << endl;
    }
}
```

Turno 2

MVP

```
#include <iostream>
#include <vector>
using namespace std;

struct Player {
    string name;
    double score;      // mean score
};

struct Team {
    string tname;
    Player mvp;        // most valuable player
};

typedef vector <Team> League;

// Pre: en la entrada hay una secuencia que representa la info de los equipos
//      tal y como describe el enunciado
// Post: devuelve una League a partir de los datos de la entrada
League read_data() {
    int n;
    cin >> n;
    League lg(n);
    for (int i = 0; i < n; ++i) {
        cin >> lg[i].tname >> lg[i].mvp.name >> lg[i].mvp.score;
    }
    return lg;
}

int main() {
    League lg = read_data();
    char c;
    double s;
    while (cin >> c >> s) {
        cout << "-- begin query" << endl;
        for (int i = 0; i < lg.size(); ++i) {
            if (lg[i].tname[0] == c and lg[i].mvp.score > s) {
                cout << lg[i].mvp.name << endl;
            }
        }
        cout << "-- end query" << endl;
```

```
    }
}
```

Posiciones acumulación (2)

```
#include <iostream>
#include <vector>
using namespace std;

// Pre: en la entrada hay una secuencia de v.size() enteros
// Post: el vector v almacena los enteros de la entrada en el mismo
//           orden en el que aparecen
void leer_vector(vector<int>& v) {
    int n = v.size();
    for (int i = 0; i < n; ++i) cin >> v[i];
}

// Pre: los elementos de v son no negativos
// Post: retorna la cantidad de posiciones de acumulacin de v
int num_cumulos(const vector<int>& v) {
    int contador = 0;
    int n = v.size();
    for (int i = 0; i < n; ++i) {
        int suma = 0;
        int j = i + 1;
        while (j < n and suma < v[i]) {
            suma += v[j];
            ++j;
        }
        if (suma == v[i]) ++contador;
    }
    return contador;
}

int main() {
    int n;
    while (cin >> n) {
        vector<int> v(n);
        leer_vector(v);
        cout << num_cumulos(v) << endl;
    }
}
```

Número de filas diferentes

```
#include <iostream>
```

```cpp
#include <vector>
using namespace std;

typedef vector<int> Row;
typedef vector<Row> Matrix;

//reads a matrix from input
void read_mat(Matrix& mat) {
    int m = mat.size();
    int n = mat[0].size();
    for (int i = 0; i < m; ++i)
        for (int j = 0; j < n; ++j)
            cin >> mat[i][j];
}

//pre: i and j are two valid row indexes of mat
//post: returns true if rows i and j are the same
//      Returns false otherwise
bool are_equal(const Matrix& mat, int i, int j) {
    int n = mat[0].size();
    for (int k = 0; k < n; ++k)
        if (mat[i][k] != mat[j][k]) return false;
    return true;
}

//pre: i is valix index row of mat
//post: returns true when row i is different from  any previous one
//      Returns false otherwise
bool is_new_row(const Matrix& mat, int i) {
    for (int j = 0; j < i; ++j)
        if (are_equal(mat, j, i)) return false;
    return true;
}

int main() {
    int m, n;
    while (cin >> m >> n) {
        Matrix mat(m, Row(n));
        read_mat(mat);
        int counter = 0;
        for (int i = 0; i < m; ++i)
            if (is_new_row(mat,i)) ++counter;
        cout << counter << endl;
    }
}
```