MATRICES -2

Implementar mediante JFlex y CUP un interprete que realice cálculos básicos con vectores y matrices de números reales. El siguiente es un ejemplo de programa en este lenguaje:

```
a = {{1,2,3},{4,5,6},{7,8,9}};
print(a);
b1 = {1,2,3};
b2 = {4,5,6};
c = b1 : b2;
d = b1 * b2:3:4;
print(e);
f = {b, b+{3,3,3},{}:{7:8}:{9}}
print(f + a);
```

Las matrices son siempre de dos dimensiones *n* x *m*, siendo $1<n<1000$ & $1<m<1000$. Se acepta expresamente el vector y la matriz vacía que se escriben respectivamente como:

```
v = {};
m = {{}};
```

Se definen los operadores entre escalares, vectores y matrices. Se introducen los siguientes operadores entre vectores: (ver ejemplos en los casos de prueba)

- Concatenación: (Operador :) Añade los elementos de un vector al final de otro. Este operador tiene la máxima prioridad. Puede concatenar dos vectores o bien un vector y un número, o bien una matriz y un vector en cuyo caso añade una columna al final (si las dimensiones son adecuadas).
- Producto vectorial: (Operador *) Multiplica dos vectores de dimensiones n y m respectivamente y obtiene una matriz de *n* x *m*. También funciona multiplicando un vector por una matriz, o una matriz por un vector. El resultado es siempre una matriz.

Las operaciones aritméticas entre vectores y matrices requieren a su vez condiciones conocidas: Por ejemplo la suma de vectores o matrices requiere que ambos operandos tengan las mismas dimensiones, el producto requiere que el número columnas de la primera matriz coincida con el numero de filas de la segunda. Si se trata de un vector por una matriz, debe coincidir la dimensión del vector por el número de filas, y si es de una matriz por un vector, el numero de columnas de la matriz con la dimensión del vector.

Se define el concepto de "subvector" y "submatriz", que aplicado a un vector o a una matriz definida previamente permite obtener una parte superior izquierda de la matriz original. (Ver casos de prueba). Finalmente, se introduce el concepto de "matriz dispersa", entendiendo como tal una matriz en la que solo se definen algunos valores, rellenando el resto con ceros. (Ver casos de prueba).

Se proporciona un analizador léxico implementado mediante *JFlex*, en el fichero `Matrices.flex`, la clase `Matrices.java` que contiene la implementación de las operaciones entre matrices, métodos auxiliares[1] y el método `main`. La lista de funciones disponibles se limita a los definidos en el analizador léxico.

Para compilar y ejecutar este interprete se usarán la secuencia:

```
cup   Matrices.cup
jflex Matrices.flex
javac Matrices.java
java  Matrices <entrada>
```

Para la corrección de este ejercicio se entrega SOLAMENTE el fichero `Matrices.cup` por lo que cualquier modificación en cualquier otro fichero no se tendrá en cuenta al corregir.

Se proporcionan diversos casos de prueba[2], con diversos aspectos que deben controlarse en la implementación.

---

[1] Para realizar la salida debe emplearse necesariamente los métodos `print(double[])`y `print(double[][])` de la clase `Matrices`,para vectores y matrices respectivamente. Para la lectura puede utilizarse las clases `ArrayList<Doble>` y `ArrayList<ArrayList<Double>>`. Se proporcionan métodos de conversión entre ambas representaciones.

[2] Los casos de prueba que se muestran son solo un ejemplo, a los que llamaremos casos de prueba públicos. Para evitar implementaciones "ad hoc" se probaran otros casos de prueba similares a estos, pero que no se proporcionan. A estos casos se les llama casos de prueba privados. Para superar cada prueba hay que superar ambos conjuntos de casos de prueba.

Caso 1. Declaración e impresión de vectores y matrices

Entrada	*Salida*

`a = {{1,2,3},{4,5,6},{7,8,9}};`
`print(a);`
`b = {1,2,3};`
`print(b);`
	`1,00 2,00 3,00`
	`4,00 5,00 6,00`
	`7.00 8,00 9,00`
	`1,00 2,00 3,00`

`a = {{1,2},{-4,5},{7,-8}};`
`print(a);`
`b = {1,2,-3,4,5};`
`print(b);`
	`1,00  2,00`
	`-4,00  5,00`
	`7,00  -8,00`
	`1,00  2,00  -3,00  4,00  5,00`

Caso 2. Operaciones con vectores. Concatenación

Entrada	*Salida*

`b = {1,2,3};`
`c = {4,5};`
`print(b:c);`
	`1,00 2,00 3,00 4,00 5,00 6,00`

`b = {1,2}:{3,4}:{5,6};` `print(b);`	`1,00 2,00 3,00 4,00 5,00 6,00`
`b = {1,2}:3:{4} + 4:{3,2,1};` `print(b);`	`5,00 5,00 5,00 5,00`
`a = {{1,2,3},{4,5,6}};` `print(a:{7,8});`	`1,00  2,00  3,00  7,00` `4,00  5,00  6,00  8,00`

Caso 3. Sustitución de variables en las definiciones de vectores y matrices

Entrada	*Salida*

`b = {1,2,3};`
`a = {b,b+{3,3,3},{7,8,9}};`
`print(a);`
	`1,00 2,00 3,00`
	`4,00 5,00 6,00`
	`7.00 8,00 9,00`

`a = {1,2};`
`b = a:a:a;`
`print(b);`
	`1,00 2,00 1,00 2,00 1,00 2,00`

`a = {1,2};`
`b = {3,4};`
`c = { a:b+b:a,{1,2,3,4}+b:a };`
`print(c);`
	`4,00 6,00 4,00 6,00`
	`4,00 6,00 4,00 6,00`

Caso 4. Operaciones entre vectores y matrices

Entrada	*Salida*

`a = {{1,2},{4,5},{7,8}};`
`b = {1,2,3};`
`print (b*a);`
	`30,00  36,00`

`a = {{1,2,3},{4,5,6}};`
`b = {2,3,4};`
`print (a*b);`
	`20,00`
	`47,00`

`a = {{1,2,3},{4,5,6},{7,8,9}};`
`b = {2,3,4};`
`print (b*a*b);`
	`477,00`

`b = {2,3,4};` `print (b*b);`	`29,00`

Caso 5. Submatrices. Para definir una submatriz se añade entre paréntesis el rango después del identificador de la matriz. Es necesario que la submatriz haga referencia a variables que ya están definidas. Para ello definiremos el tamaño de la submatriz (filas y columnas) indicándolo entre paréntesis a continuación del nombre de la variable.

Entrada	*Salida*
`a = {{1,2,3},{4,5,6},{7,8,9}};` `print(a(2,2));`	`1,00 2,00` `4,00 5,00`

`a = {{1,2,3},{4,5,6},{7,8,9}};`
`b = a(3,2);`
`print(b);`
	`1,00 2,00`
	`4,00 5,00`
	`7,00 8,00`

`a = {{1,2,3},{4,5,6},{7,8,9}};`
`b = a(2,2);`
`print(b*a(2,3));`
	`9,00 12,00 15,00`
	`24,00 33,00 42,00`

`a = {{1,2,3},{4,5,6},{7,8,9}};` `b = a(4,2);`	`ERROR7: se quieren usar mas filas de las disponibles`
`a = {{1,2,3},{4,5,6},{7,8,9}};` `b = a(2,4);`	`ERROR8: se quieren usar mas columnas de las disponibles`

Caso 6. Matrices dispersas. Se desea incorporar también la declaración de un tipo de matrices dispersas. Queremos declarar matrices que pueden tener menos filas de las indicadas (que se rellenarán con 0) o con columnas que tampoco lleguen al número de columnas indicado (que se rellenarán con 0).

El uso de esta declaración de matrices dispersas está reservado al momento en que se declaran variables. Las dimensiones que se guardarán de forma efectiva en la tabla de símbolos se indicará con dos números entre paréntesis (filas y columnas).

1	2	3	
4	5		
6			

Sería este en tamaño 4x4

1	2	3	0
4	5	0	0
6	0	0	0
0	0	0	0

Entrada	*Salida*
`a(2,3) = {{1,2,3},{4,5}};` `print(a);`	`1,00 2,00 3,00` `4,00 5,00 0,00`

`a(3,3) = {{1,2,3},{4,5}};`
`print(a);`
	`1,00 2,00 3,00`
	`4,00 5,00 6,00`
	`0,00 0,00 0,00`

`a(4,4) = {{1,2,3},{4,5}};`
`print(a);`
	`1,00 2,00 3,00 0,00`
	`4,00 5,00 0,00 0,00`
	`6,00 0,00 0,00 0,00`
	`0,00 0,00 0,00 0,00`

Caso 7. Si las dimensiones de la matriz recibida son mayores que las de la matriz que se quiere guardar, se eliminarán las filas y columnas que sobren

Entrada	*Salida*
`a(2,4) = {{1,2,3,4,5,6},{7,8,9,1}};` `print(a);`	`1,00 2,00 3,00 4,00` `7,00 8,00 9,00 1,00`
`a(2,3) = {{1,2,3,4,5,6},{7},{8,9,4,5}};` `print(a);`	`1,00 2,00 3,00` `7,00 0,00 0,00`

Caso 8. Combinación de ambas funcionalidades:

Entrada	*Salida*
`a(2,3) = {{1,2,3,4,5,6},{7},{8,9,4,5}};` `print(a(2,2));`	`1,00 2,00` `7,00 0,00`
`a(3,3) = {{1,2,3,4,5,6},{7},{8,9,4,5}};` `print(a(2,3)*a(3,2));`	`39,00 29,00` `7,00 14,00`