Exercise 2

Michael has sent Richard a list of PC components and their prices in a CSV file.

1. Richard cannot import this file into Excel because it was created on a European PC where the convention for separators and decimal markers is different from that on a UK or US PC. Convert it so that he can – that is, recreate it with separators changed from ';' to ',' and the decimal markers changed from ',' to '.'.
2. Richard would like the prices in the file to be specified in pounds rather than euros. Convert them, assuming £1 = €1.10.

The CSV file is Exercise2.csv.

This exercise requires Dyalog 16.0.

Hints

`⎕CSV` can be used to read the file into the workspace and write it out again formatted differently.

By default, `⎕CSV` expects ',' separators and '.' decimal markers in the file it reads so it won't be able to process it without overriding these using variant options.

By default, `⎕CSV` will assume all fields in a CSV file contain character data and won't convert to numeric data without specifying the column data types in the right argument.

The manual pages are attached.

Once the data has been read in, writing the new file out again will be trivial. If the data is read into `d` then the following will write it out in the expected format:

```
d ⎕CSV 'new.csv'
```

`⎕CSV` will not overwrite an existing file by default. If it already exists, either use `⎕NDELETE` to erase it first, or:

```
d (⎕CSV⍠'Overwrite' 1) 'new.csv'
```

Comma Separated Values {R}←{X} ⎕CSV Y

This function imports and exports Comma Separated Value (CSV) data.

Monadic `⎕CSV` imports data from a CSV file or converts data from CSV format to an internal format. Dyadic `⎕CSV` exports data to a CSV file or converts data from internal format to a CSV format.

Internal Format

Arrays that result from importing CSV data or arrays that are suitable for exporting as CSV data are represented by 3 possible structures:

- A table (a matrix whose elements are character vectors or scalars, or numbers).
- A vector, each of whose items contain field (column) values. Character field values are character matrices; numeric field values are numeric vectors.
- A vector, each of whose items contain field (column) values. Character field values are vectors of character vectors; numeric field values are numeric vectors.

Note that when importing CSV data, all fields are assumed to be character fields unless otherwise specified (see *Column Types* below). A field that contains only "numbers" will not be converted to numeric data unless specified as being numeric.

Monadic ⎕CSV

```
R←⎕CSV Y
```

`Y` is an array that specifies just the source of the CSV data (see below) or a 1,2,3 or 4-element vector containing:

`[1]`	Source of CSV Data (see below)
`[2]`	Description of the CSV data (see below)
`[3]`	Column Types (see below)
`[4]`	Boolean header row indicator (see below)

The first record in a CSV file is often a list of column labels. If present this *header row* is treated differently from other records. It is assumed to contain character data (labels) regardless of `Y[3]` and is returned separately in the result.

Source - may be one of:

- a character vector or scalar containing a file name
- a native tie number
- a character vector or scalar containing CSV data with embedded newline characters[1]
- a vector of character vectors and/or scalars containing CSV data with implicit newlines after each character vector or scalar

Description may be one of:

- a character vector specifying the file encoding such as `'UTF-8'` (see *File Encodings* on page 442). This applies when `Y[1]` is a file name or tie number. If omitted or empty, the file encoding is deduced (see below).
- a character scalar `'S'` (simple) or `'N'` (nested). This applies when `Y[1]` is a character array containing CSV data. The default is `'N'`.

Column Types

This is a scalar numeric code or vector of numeric codes that specifies the field types from the list below. If *Column Types* is zilde or omitted, the default is 1 (all fields are character).

`0`	The field is ignored.
`1`	The field contains character data.
`2`	The field is to be interpreted as being numeric. Empty cells and cells which cannot be converted to numeric values are not tolerated and cause an error to be signalled.
`3`	The field is to be interpreted as being numeric but invalid numeric vales are tolerated. Empty fields and fields which cannot be converted to numeric values are replaced with the Fill variant option (default 0).
`4`	The field is to be interpreted numeric data but invalid numeric data is tolerated. Empty fields and fields which cannot be converted to numeric values are returned instead as character data; this type is disallowed when variant option Invert is set to 1.

Note that if *Column Types* is specified by a scalar 4, all numeric data in all fields will be converted to numbers.

[1]Note that when the `Y[1]` is a character vector or scalar containing CSV data, `Y[2]` must be specified as `'S'`. Otherwise `Y[1]` will be interpreted as a file name.

Variant options

The following variant options are accepted:

Name	Meaning	Default
Invert	0, 1 or 2 (see below)	`0`
Separator	the field separator, any single character. If Widths is other than `⍬`, Separator is ignored.	`','`
Widths	a vector of numeric values describing the width (in characters) of the corresponding columns in the CSV source, or `⍬` for variable width delimited fields	`⍬`
Decimal	the decimal mark in numeric fields - one of `'.'` or `','`	`'.'`
Thousands	the thousands separator in numeric fields, which may be specified as an empty character vector (meaning no separator is defined) or a character scalar	`''`
Trim	a Boolean specifying whether undelimited/unescaped whitespace is trimmed at the beginning and end of fields	`1`
Ragged	a Boolean specifying whether records with varying numbers of fields are allowed; see notes below	`0`
Fill	the numeric value substituted for invalid numeric data in columns of type 3	`0`
Records	the maximum number of records to process or 0 for no limit	`0`

Other options defined for export are also accepted but ignored.

Invert

This option specifies how the CSV data should be returned as follows:

0	A table (a matrix whose elements are character vectors or scalars or numbers).
1	A vector, each of whose items contain field (column) values. Character field values are character matrices; numeric field values are numeric vectors.
2	A vector, each of whose items contain field (column) values. Character field values are vectors of character vectors; numeric field values are numeric vectors.

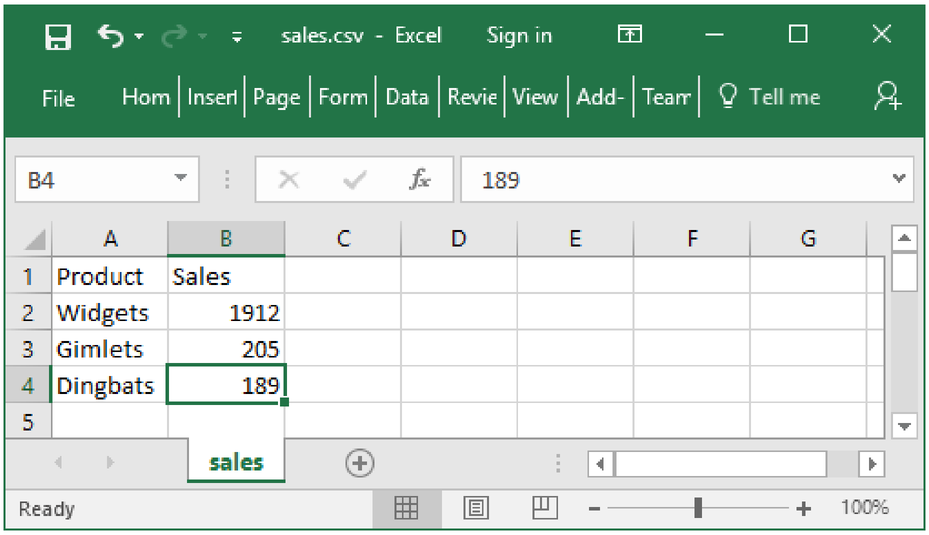
The result R contains the imported data.

If Y[4] does not specify that the data contains a header then R contains the entire data in the form specified by the Invert variant option.

If Y[4] does specify that the data contains a header then R is a 2-element vector where:

- R[1] is the imported data excluding the header.
- R[2] is a vector of character vectors containing the header record.

Examples



	A	B
1	Product	Sales
2	Widgets	1912
3	Gimlets	205
4	Dingbats	189
5		

```
      ⊃⎕NGET CSVFile←'c:\Dyalog16.0\sales.csv'
┌→──────────────────────────────────────────────┐
│Product,Sales                                  │
│             Widgets,1912                      │
│                         Gimlets,205           │
│                                    Dingbats,189│
│                                               │
└───────────────────────────────────────────────┘

      ⎕CSV CSVFile
┌→───────────────────┐
↓ ┌→──────┐  ┌→────┐ │
│ │Product│  │Sales│ │
│ └───────┘  └─────┘ │
│ ┌→──────┐  ┌→───┐  │
│ │Widgets│  │1912│  │
│ └───────┘  └────┘  │
│ ┌→──────┐  ┌→──┐   │
│ │Gimlets│  │205│   │
│ └───────┘  └───┘   │
│ ┌→───────┐ ┌→──┐   │
│ │Dingbats│ │189│   │
│ └────────┘ └───┘   │
└∊───────────────────┘
      ⎕CSV CSVFile'' ⍬ 1 ⍝ Header row
┌→────────────────────────────────────────────┐
│ ┌→──────────────────┐ ┌→──────────────────┐ │
│ ↓ ┌→──────┐  ┌→───┐ │ │ ┌→──────┐ ┌→────┐ │ │
│ │ │Widgets│  │1912│ │ │ │Product│ │Sales│ │ │
│ │ └───────┘  └────┘ │ │ └───────┘ └─────┘ │ │
│ │ ┌→──────┐  ┌→──┐  │ └∊──────────────────┘ │
│ │ │Gimlets│  │205│  │                       │
│ │ └───────┘  └───┘  │                       │
│ │ ┌→───────┐ ┌→──┐  │                       │
│ │ │Dingbats│ │189│  │                       │
│ │ └────────┘ └───┘  │                       │
│ └∊──────────────────┘                       │
└∊────────────────────────────────────────────┘
```

```
      ⎕CSV CSVFile''(1 2)1 ⍝ Fields are Char,Num
┌→────────────────────────────────────────────┐
│ ┌→───────────────────┐ ┌→──────────────────┐│
│ ↓ ┌→──────┐          │ │ ┌→──────┐ ┌→────┐ ││
│ │ │Widgets│  1912    │ │ │Product│ │Sales│ ││
│ │ └───────┘          │ │ └───────┘ └─────┘ ││
│ │ ┌→──────┐          │ └∊──────────────────┘│
│ │ │Gimlets│  205     │                      │
│ │ └───────┘          │                      │
│ │ ┌→───────┐         │                      │
│ │ │Dingbats│ 189     │                      │
│ │ └────────┘         │                      │
│ └∊───────────────────┘                      │
└∊────────────────────────────────────────────┘
      (⎕CSV⍠'Invert' 1)CSVFile'' (1 2) 1     ⍝ Invert 1
┌→──────────────────────────────────────────────────────┐
│ ┌→────────────────────────────┐ ┌→──────────────────┐ │
│ │ ┌→───────┐ ┌→────────────┐  │ │ ┌→──────┐ ┌→────┐ │ │
│ │ ↓Widgets │ │1912 205 189 │  │ │ │Product│ │Sales│ │ │
│ │ │Gimlets │ └~────────────┘  │ │ └───────┘ └─────┘ │ │
│ │ │Dingbats│                  │ └∊──────────────────┘ │
│ │ └────────┘                  │                       │
│ └∊────────────────────────────┘                       │
└∊──────────────────────────────────────────────────────┘
      ⊃(⎕CSV⍠'Invert' 2)CSVFile'' (1 2) 1     ⍝ Invert 2
┌→───────────────────────────────────────────────────┐
│ ┌→─────────────────────────────────┐ ┌→───────────┐│
│ │ ┌→──────┐ ┌→──────┐ ┌→───────┐   │ │1912 205 189││
│ │ │Widgets│ │Gimlets│ │Dingbats│   │ └~───────────┘│
│ │ └───────┘ └───────┘ └────────┘   │               │
│ └∊─────────────────────────────────┘               │
└∊───────────────────────────────────────────────────┘
```

Notes

- When `Y` specifies just the source of the CSV data, it does not need to be enclosed or ravelled to create a 1-element vector.
- `Y[2]`, the description of the source, distinguishes an otherwise ambiguous character vector source (which could contain either CSV data or a file name). The other source forms are unambiguous but the description, when given, must still match the given source type.
- Tab-separated fields may be imported by specifying `'Separator' (⎕UCS 9)`.
- Fields containing embedded new lines are supported (they must, of course, appear in quotes). On import, line endings are always converted to a single line feed character.

- If Ragged is not set then all records must have the same number of fields (character delimited format) or same number of characters (fixed width field format).
- If Ragged is set:
    - The expected number of columns must be specified using the Widths variant option and/or the column types in `Y[3]`.
    - In character delimited format, all processed records are implicitly extended or truncated as required so that they contain the expected number of fields; implicitly added fields will be empty.
    - In fixed width format, all processed records are implicitly extended with spaces or truncated as required so that they contain as many characters as are specified in the Widths option declaration.

File handling

Data may be read from a named file or a tied native file. A tied native files may be read in sections by repeatedly invoking `⎕CSV` for a specified maximum number of records (specified by the Records variant) until no more data is read.

In all cases the files must contain text using one of the supported encodings. See *File Encodings* on page 442. The method used to determine the file encoding is as follows:

- If a Byte Order Mark (BOM) is encountered at the start of the file, it is used regardless of `Y[2]` (if specified). Note, however, that the BOM can only be encountered if the file is read from the start - specifically, if a native file is read in sections, any BOM present will only be encountered when the first section is read.
- Otherwise, the file will be read and decoded according to the file encoding in `Y[2]` if specified.
- Otherwise:
    - Native files will be decoded as if `'UTF-8'` had been specified.
    - Files specified by name will be examined and the likely file encoding will be deduced using the same heuristics performed by `⎕NGET`.

Note also:

- Native files are read from the current file position. On successful completion, the file position will be at the first unprocessed character (end of file if the Records variant option is not specified). If an error is signalled the file position is undefined.
- The result does not report the file encoding or line ending type as it does with `⎕NGET`. If this information is required then it must be obtained by other means.

Dyadic ⎕CSV

`{R}←X ⎕CSV Y`

The left argument `X` is either:

- a matrix or a vector of vectors/matrices containing the data to be converted to CSV format.
- or a 2-element vector containing a matrix or vector of vectors/matrices containing the data to be converted to CSV format, and a vector of character vectors containing the header record.

`Y` is a 1 or 2-element vector containing:

`[1]`	Destination of CSV Data (see below)
`[2]`	Description of the CSV data (see below)

Destination - may be one of:

- a character vector or scalar containing a file name
- a native tie number
- an empty character vector, indicating that the CSV data is to be returned in the result `R`

Description may be:

- a character vector specifying the file encoding such as `'UTF-8'` (see *File Encodings* on page 442). This applies when `Y[1]` is a file name or tie number. If omitted or empty, the file encoding defaults to `'UTF-8'`.
- a character scalar `'S'` (simple) or `'N'` (nested). This applies when `Y[1]` is empty. The default is `'S'`.

Variant options

The following variant options are accepted:

Name	Meaning	Default
Overwrite	a Boolean which specifies, when creating a named file which already exists, whether to overwrite it (1) or signal an error (0)	`0`
Separator	the field separator, any single character. If Widths is other than `⍬`, Separator is ignored.	`','`
Widths	a vector of numeric values describing the width (in characters) of the corresponding columns in the CSV source, or `⍬` for variable width delimited fields	`⍬`
Decimal	the decimal mark in numeric fields - one of `'.'` or `','`	`'.'`
Thousands	the thousands separator in numeric fields, which may be specified as an empty character vector (meaning no separator is defined) or a character scalar	`''`
Trim	a Boolean specifying whether whitespace is trimmed at the beginning and end of character fields	`1`
LineEnding	the line ending sequence - see *Line separators: on page 443*	(13 10) on Windows; 10 on other platforms

Other options defined for import are also accepted but ignored.

If `Y` specifies that the CSV data is written to a file then `R` is the number of bytes (not characters) written, and is shy.

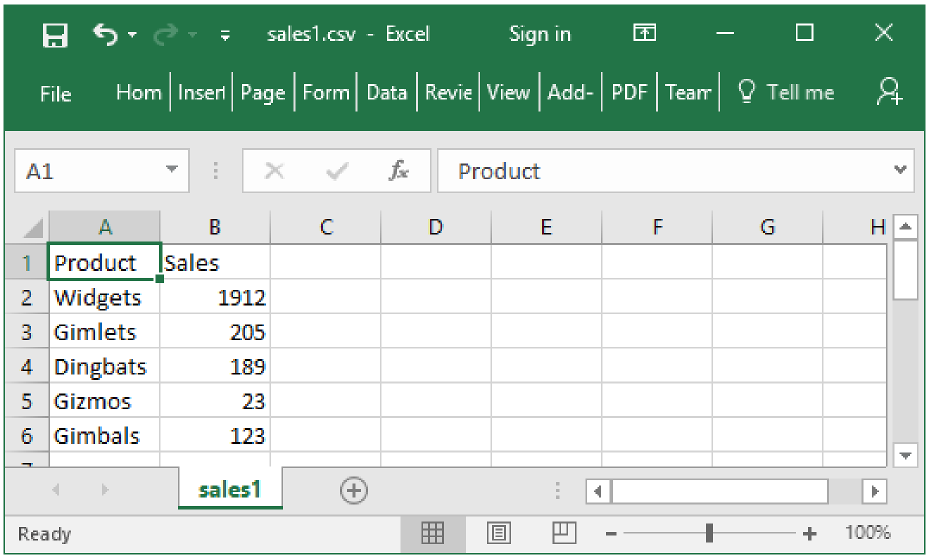
Otherwise, `R` is the CSV data in the format specified in Y, and is not shy.

Examples

```
      CSVFile←'c:\Dyalog16.0\sales.csv'
      ⎕←DATA HDR←⎕CSV CSVFile''(1 2)1
┌→─────────────────────────────────────────────┐
│ ┌→───────────────────┐ ┌→───────────────────┐ │
│ ↓ ┌→───────┐         │ │ ┌→──────┐ ┌→────┐ │ │
│ │ │Widgets │ 1912    │ │ │Product│ │Sales│ │ │
│ │ └────────┘         │ │ └───────┘ └─────┘ │ │
│ │ ┌→───────┐         │ └∊──────────────────┘ │
│ │ │Gimlets │ 205     │                       │
│ │ └────────┘         │                       │
│ │ ┌→────────┐        │                       │
│ │ │Dingbats │ 189    │                       │
│ │ └─────────┘        │                       │
│ └∊───────────────────┘                       │
└∊─────────────────────────────────────────────┘
     DATA⍪←'Gizmos' 23
     DATA HDR ⎕CSV''
┌→───────────┐
│Product,Sales│
│             │
│Widgets,1912 │
│             │
│Gimlets,205  │
│             │
│Dingbats,189 │
│             │
│Gizmos,23    │
│             │
└─────────────┘
      CSVFile1←'c:\Dyalog16.0\sales1.csv'
      ⎕←DATA HDR ⎕CSV CSVFile1

67
      DATA⍪←'Gimbals' 123
      ⎕←DATA HDR ⎕CSV CSVFile1
FILE NAME ERROR: Unable to create file ("The file
exists.")
      ⎕←DATA HDR ⎕CSV CSVFile1
     ∧
      ⎕←DATA HDR(⎕CSV⍠'Overwrite' 1)CSVFile1

80
```



	A	B
1	Product	Sales
2	Widgets	1912
3	Gimlets	205
4	Dingbats	189
5	Gizmos	23
6	Gimbals	123

Notes

- When `Y` contains only the destination of the CSV data (i.e. omits the description in its second element) it does not have to be enclosed to form a single element vector.
- Native files are written from the current file position. On successful completion, the file position will be at the end of the written data. If an error is signalled the amount of data written is undefined.
- If the file encoding specifies that a BOM is required and output is to a native file, it will only be written if the file position is initially at 0 - that is, the start of the file is being written.
- When fixed width fields are written, character data shorter than the specified width is padded with spaces to the right and character data longer than the specified width signals an error. Numeric data is converted to character data as far as possible so that it fits into the specified width. If this is not possible, an error is signalled.
- Tab-separated fields may be exported by specifying `'Separator' (⎕UCS 9)`.
- Fields containing a single embedded new line are supported. On export, line feed characters are mapped back to the defined line ending sequence.