

# Lab 11: CSV to HTML Table Renderer

## CS1234: Small-Scale Application Development

### Background

Comma-Separated Values (CSV) is one of the simplest and most widely used formats for storing tabular data. Spreadsheet tools and databases often export data in CSV format because it is easy to read and process.

HTML tables are commonly used to display tabular data on web pages. In this lab, you will write a C program that reads data from a CSV file and renders it as an HTML table. The appearance of the table will not be hardcoded; instead, it will be controlled using a separate configuration file.

### Problem Statement (5 marks)

You are required to implement a **CSV to HTML table renderer** in C.

Your program will take **one command-line argument**: the configuration file (for example, `input##.txt`) containing rendering properties. The file `data.csv` will be present in the lab folder, and your program should read it from there.

The program must read the contents of `data.csv`, read the configuration properties from the given configuration file, and print the corresponding HTML table to `STDOUT`.

The configuration file controls the rendering of the table, including:

- border size
- table width
- border color
- cell alignment
- alternate row color
- whether the first row should be rendered as a header row
- optional table caption

You may assume that **all input files are valid and error-free**. You do not need to handle malformed CSV, malformed configuration lines, invalid property values, or file-format errors.

### Input Format

Your program accepts the following command-line argument:

```
./a.out input##.txt
```

In the evaluation test cases, `data.csv` will remain fixed and only the configuration file will vary.

#### 1. CSV File

The CSV file contains the table contents.

Rules:

- The CSV file is guaranteed to contain at least one row.

- Each line represents exactly one row of the table.
- Fields in a row are separated by commas ,.
- No field will contain commas inside it.
- No field will contain double quotes.
- No field spans multiple lines.
- All rows will contain the same number of fields.
- Leading and trailing spaces inside fields should be preserved as part of the field.
- Empty lines will not appear in the CSV file.
- No empty field will appear in the CSV file.

## 2. Configuration File

The configuration file contains one property per line in the following format:

`key=value`

Additional rules:

- There will be no spaces before or after =.
- Empty lines will not appear in the configuration file.
- Each key appears at most once.
- The configuration file may omit some keys.

Supported keys are:

- `border_size`
- `table_width`
- `border_color`
- `cell_alignment`
- `row_color`
- `has_header`
- `caption`

### Supported values

- `border_size`: a non-negative integer
- `table_width`: a string such as 50%, 75%, 100%
- `border_color`: a single word such as `black`, `blue`, `green`, `red`
- `cell_alignment`: one of `left`, `center`, `right`
- `row_color`: a single word such as `lightgray`, `lightyellow`, `aliceblue`
- `has_header`: either `yes` or `no`
- `caption`: any text after =

## Default values

If a property is not present in the configuration file, use the following default:

- `border_size=1`
- `table_width=100%`
- `border_color=black`
- `cell_alignment=left`
- `row_color=` (empty, meaning no alternate row color should be applied)
- `has_header=yes`
- `caption=` (empty, meaning the `<caption>` tag should not be printed at all)

## Output Format

Print the HTML table to `STDOUT`.

### Opening table tag

Print the opening `<table>` tag in **exactly** the following format and attribute order:

```
<table border="<border_size>" width="<table_width>" style="border-color:<border_color>; text-align:<cell_alignment>;">
```

### Caption

If `caption` is non-empty, print:

```
<caption>...</caption>
```

If `caption` is empty or missing, omit the caption entirely. In particular, do **not** print `<caption></caption>`.

### Rows

Print exactly one row per line.

- If `has_header=yes`, the first CSV row must be printed using `<th>` tags.
- If `has_header=no`, the first CSV row must still be printed, but using `<td>` tags like a normal data row.
- All remaining rows must be printed using `<td>` tags.
- If `row_color` is non-empty, apply it to the 1st, 3rd, 5th, ... data rows.
- A **data row** means any row printed using `<td>` tags. So, if `has_header=yes`, coloring starts from the second CSV row; otherwise, coloring starts from the first CSV row.

### Header row format

```
<tr><th>field1</th><th>field2</th>...</tr>
```

### Data row format

```
<tr><td>field1</td><td>field2</td>...</tr>
```

## Alternate colored data row format

If the row is one of the alternate data rows that must receive `row_color`, print:

```
<tr style="background-color:<row_color>;"><td>field1</td><td>field2</td>...</tr>
```

## Closing tag

Print:

```
</table>
```

## Formatting notes

- Print the output exactly in the specified format.
- Print each tag sequence exactly as shown above.
- Do not print extra spaces inside row tags.
- Print each row on its own line.
- Do not print blank lines anywhere in the output.

## Sample Test Cases

### Common CSV File

**data.csv**

```
Name,Roll,Marks
Asha,CS001,91
Rahul,CS002,88
Meena,CS003,95
```

### Test Case 0

#### Input configuration file

**input01.txt**

```
border_size=2
table_width=80%
border_color=blue
cell_alignment=center
row_color=lightgray
has_header=yes
caption=Class Report
```

#### Output

```
<table border="2" width="80%" style="border-color:blue; text-align:center;">
<caption>Class Report</caption>
<tr><th>Name</th><th>Roll</th><th>Marks</th></tr>
<tr style="background-color:lightgray;"><td>Asha</td><td>CS001</td><td>91</td></tr>
<tr><td>Rahul</td><td>CS002</td><td>88</td></tr>
<tr style="background-color:lightgray;"><td>Meena</td><td>CS003</td><td>95</td></tr>
</table>
```

## Rendered in a browser

| Name  | Roll  | Marks |
|-------|-------|-------|
| Asha  | CS001 | 91    |
| Rahul | CS002 | 88    |
| Meena | CS003 | 95    |

## Test Case 1

### Input configuration file

input02.txt

```
border_size=4
table_width=60%
border_color=green
cell_alignment=left
row_color=lightyellow
has_header=no
caption=Student Data
```

### Output

```
<table border="4" width="60%" style="border-color:green; text-align:left;">
<caption>Student Data</caption>
<tr style="background-color:lightyellow;"><td>Name</td><td>Roll</td><td>Marks</td></tr>
<tr><td>Asha</td><td>CS001</td><td>91</td></tr>
<tr style="background-color:lightyellow;"><td>Rahul</td><td>CS002</td><td>88</td></tr>
<tr><td>Meena</td><td>CS003</td><td>95</td></tr>
</table>
```

## Rendered in a browser

| Name  | Roll  | Marks |
|-------|-------|-------|
| Asha  | CS001 | 91    |
| Rahul | CS002 | 88    |
| Meena | CS003 | 95    |

## Test Case 2

### Input configuration file

input03.txt

```
border_color=red
caption=
```

### Output

```
<table border="1" width="100%" style="border-color:red; text-align:left;">
<tr><th>Name</th><th>Roll</th><th>Marks</th></tr>
<tr><td>Asha</td><td>CS001</td><td>91</td></tr>
<tr><td>Rahul</td><td>CS002</td><td>88</td></tr>
<tr><td>Meena</td><td>CS003</td><td>95</td></tr>
</table>
```

## Rendered in a browser

| Name  | Roll  | Marks |
|-------|-------|-------|
| Asha  | CS001 | 91    |
| Rahul | CS002 | 88    |
| Meena | CS003 | 95    |

## Constraints

| Parameter                         | Limit          |
|-----------------------------------|----------------|
| Maximum number of rows in CSV     | 100            |
| Maximum number of columns in CSV  | 20             |
| Maximum length of one CSV line    | 512 characters |
| Maximum length of one config line | 256 characters |
| Maximum field length              | 100 characters |
| Maximum caption length            | 100 characters |

## Notes & Submission Guidelines

- Run `check.sh` to test against the public test cases.
- Run `submit.sh` to submit.
- All test inputs are guaranteed to be valid.
- Memory note: Free any dynamically allocated memory before exit.
- For easier visualization, you may run `./a.out input##.txt > out.html` and then open `out.html` in a browser to see the rendered table.

## Extras (Ungraded)

### Extension 1: Per-Column Alignment

Support config entries such as:

```
col_alignment_1=left
col_alignment_2=center
col_alignment_3=right
```

These should override the default `cell_alignment` for specific columns.

### Extension 2: HTML Escaping

Replace special characters in cell contents as follows before printing:

- `&` → `&amp;`
- `<` → `&lt;`
- `>` → `&gt;`