# ILFP Assignment 2
# Translation between iCalender and CSV

The goal of this assignment is to implement a module which provides functions to parse iCalender and CSV records, and consequently facilitates translation between them.

## 1 Problem Definition

iCalender is the standard file format for sharing calenders. As an example, we have shared the academic calender in iCalender format. You may import it into owncloud or any other calender app. Details of iCalender format for has been provided in section 3.

However the iCalender format is not suitable for analyzing the data. Suppose you want to analyze all the events in an iCalender file. Then you may prefer loading the events in a spreadsheet. Such a tabular representation provides easy access to operations like sorting or filtering based on some column value. The standard approach is to convert the data to comma-separated values (CSV) format, which is widely supported by spreadsheets and database applications.

Implement the functions to allow conversion between events in iCalender format, list of event records in ML and event records in CSV format. You are expected to implement a module which has the following functions

**ical2reclist**
: Parse events from the input iCalender file. Output is a list of records, where each record is a calender event.

**csv2reclist**
: Parse events from the input CSV file. Output is a list of records, where each record is a calender event.

**reclist2ical**
: Given an input list of event records, write the events in iCalender format to an output file.

**reclist2csv**
: Given an input list of event records, write them in CSV format to an output file.

**ical2csv**
: Convert the events given in iCalender format from the input file to CSV format and write the result to an output file.

The following points must be kept in mind while writing these functions

1. You only need to handle the event components, i.e. you can safely ignore all other components if present in the input iCalender format. The only exception is alarm component which may be present as part of an event component.

2. Make sure you provide the necessary header and tail when writing the events to a file in iCalender format. You should be able to import the output file in any calendar app.

3. The iCalender format only allows 75 characters per line. You will need to implement a folding method in order to parse records from iCalender format. Similarly while writing output in iCalender format, you will need an unfolding method to break the content lines when they exceed 75 characters.

4. All record fields should have string type including those meant for date-time values.

5. The first line of CSV file will provide the field name of the records.

6. You should be able to import the CSV output file in any spreadsheet[1].

## 2 Existing Module for File Processing

You may reuse the functions provided in fileIO.sml for this assignment. This file contains useful functions which allows reading\writing files line by line.

```
signature  FILEIO =
sig
    val getclist : string -> TextIO.StreamIO.elem list
    val getclistNN : string -> char list
    val slurp : string -> TextIO.StreamIO.vector
    val readAll : string -> TextIO.StreamIO.vector
    val bite : string -> string list
    val lick : string -> string list
    val readLines : string -> string list
    val write : string * TextIO.vector -> unit
    val writeLine : string * string -> unit
    val writeLines : string * string list -> unit
    val append : string * TextIO.vector -> unit
    val appendLine : string * string -> unit
    val appendLines : string * string list -> unit
end (* sig FILEIO *)
```

## 3 iCalender Format

iCalender is defined as MIME content type text/calendar. This enables the exchange of it's data using SMTP, HTTP, memory- based clipboard or drag/drop interactions, etc,. The complete formal specification is given in RFC 5545.

However you don't need to handle the complete specification. The scope of this assignment is limited to the event components in iCalender. A simple example of the event in iCalender format is given below.

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//SabreDAV//SabreDAV 1.7.6//EN
CALSCALE:GREGORIAN
BEGIN:VEVENT
CREATED;VALUE=DATE-TIME:20141019T193057Z
```

[1]The spreadsheet application must support the CSV format given in RFC 4180. See section 3 for details.

```
UID:f5b1b7883c
LAST-MODIFIED;VALUE=DATE-TIME:20141019T193057Z
DTSTAMP;VALUE=DATE-TIME:20141019T193057Z
SUMMARY:Last day for submission of grades of PG\, Integrated M.Tech and Dua
 l Degree projects.
DTSTART;VALUE=DATE:20150630
DTEND;VALUE=DATE:20150701
CLASS:PUBLIC
END:VEVENT
END:VCALENDAR
```

The data of iCalender file is organized as individual lines of text referred as content lines. The following grammar gives the formal specification for content lines.

| | | |
|---|---|---|
| file | → | (content-line CRLF)+ |
| content-line | → | prop-name ( [;] param-name [=] param-value)* [:] prop-value CRLF |
| prop-name | → | name |
| param-name | → | name |
| name | → | [a-zA-Z0-9\-]+ |
| param-value | → | text \| quoted-text |
| text | → | ((space \| [0x21-7E]) / [",:;])* |
| quoted-text | → | ["] ((space \| [0x21-7E]) / ["])* ["] |
| prop-value | → | (space \| [0x21-7E])* |
| space | → | [ \t] |

Every content line specifies a property name, followed by zero or more parameters separated by [;] and property value separated by [:]. The property-name, parameter-name and enumerated values are case insensitive. All other values can be assumed to be case sensitive. Each content line terminates in CRLF[2]. However each line in iCalender must be within 75 characters. This requires breaking the content lines with more that 75 characters by inserting CRLF. This operation is called folding. The SUMMARY property above gives an example of folding operation. Thus you will need to apply an unfolding operation before parsing the content lines.

### 3.1 Character Set and Escaping

The property name and parameter name can only have alphanumeric characters or [-]. A property\parameter can be assigned a list of values using comma as separator. Thus we need to escape comma character using [\] if it's part of the value. The characters [:] and [;] have special meanings while parsing parameter values. Any parameter value which contains these characters must be specified as quoted text. Comma present in quoted text for parameter value does not require escaping with [\]. Double quotes should not be present in parameter values.

## 4 CSV Format

Sec:CSV

A CSV file stores tabular data in text form. It is a common file format which is supported by a large number of applications. As such it's most common use is to import\export data between different applications.

[2]CR is carriage return ('\r', ASCII code - 13) and LF is line feed ('\n', ASCII code - 10)

A general standard for CSV does not exists. A CSV file can contain any number of records separated by some kind of line break; each record consists of fields separated by some delimiter, mostly comma, semicolon or tab. Usually all records must have identical sequence of fields. RFC 4180 provides the most commonly used format. We will build our format for CSV on top of the definitions given in RFC 4180.

The following example illustrates the event description in CSV format.

```
"CREATED-param","CREATED-value","UID-param","UID-value","LAST-MODIFIED-param",
"LAST-MODIFIED-value","DTSTAMP-param","DTSTAMP-value","SUMMARY-param",
"SUMMARY-value","DTSTART-param","DTSTART-value","DTEND-param","DTEND-value",
"CLASS-param","CLASS-value" CRLF
"VALUE=DATE-TIME","20141019T193057Z",,"f5b1b7883c","VALUE=DATE-TIME",
"20141019T193057Z","VALUE=DATE-TIME","20141019T193057Z",,"Last day for
submission of grades of PG\, Integrated M.Tech and Dual Degree projects.",
"VALUE=DATE","20150630","VALUE=DATE","20150701",,"PUBLIC" CRLF
```

Note that we have explicitly denoted the line breaks by CRLF in the above example. This is to avoid confusion with the presence of line breaks used to make the example readable, but these are not expected in the actual CSV file.

The following grammar gives the formal definition of this format.

| | | |
|---|---|---|
| file | → | header CRLF record (CRLF record)* (CRLF)? |
| header | → | name ( , name)* |
| record | → | field ( , field)* |
| name | → | field |
| field | → | " (([%x20-7E] / ") \| "")* " |

The following are important points to note

1. Each line ending with CRLF represents a record. However CRLF occurring within " are part of text value.
2. The first line is the header which specifies the property names and fixes it's sequence.
3. Each iCalender property is represented using two fields, one for it's parameter(s) and the other for it's value(s).
   - If a property has multiple parameters then they should be separated using ; in the corresponding field.
   - If a property has multiple values then they should be separated using , in the corresponding field.

# References


[1] Ed. B. Desruisseaux. *Internet Calendaring and Scheduling Core Object Specification. iCalendar*. Sept. 2009. URL: `http://tools.ietf.org/html/rfc5545` (visited on 03/21/2015).

[2] Y. Shafranovich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. Oct. 2005. URL: `https://tools.ietf.org/html/rfc4180` (visited on 03/21/2015).

[3] Wikipedia. *Comma-separated values*. Mar. 2015. URL: `http://en.wikipedia.org/wiki/Comma-separated_values` (visited on 03/21/2015).

[4] Wikipedia. *iCalendar*. Mar. 2015. URL: `http://en.wikipedia.org/w/index.php?title=ICalendar&oldid=651165053` (visited on 03/21/2015).