Project 2, due Saturday, March 6 at 11:59pm Chicago time #411

A Adam Shaw STAFF
a year ago in General

Please note the Saturday deadline. It's not our intention for you to work until Saturday night, we're just trying to give you as much time as we can and still leave some space between the project2 deadline and the second exam. Also, it really won't be possible for us to grant extensions for project2, so do plan accordingly. There will be no other graded work due in the interim.

(There will be a project3 due finals week.)

Project2 extends the calendar begun in project1. While the calendar in the previous project had a current month and year, this calendar has a current day that it maintains, and indicates that day visually as well. The application furthermore acts as an up-to-the-second clock, continuously displaying the time and date at the present moment. There is a demonstration of a working project2 in the Panopto stream; the video is labeled lecture20.1 and you might want to pop over and view it at any time.

In addition to CalFormat, which persists unchanged, you will need the following new data structures for project2:

```
(define-struct Time
  ([hour : Integer] ;; from 0 to 23
   [minute : Integer]
   [second : Integer]))

(define-struct Date
  ([month : Integer] ;; 1 for January, ..., 12 for December
   [day : Integer]
   [year : Integer]))

(define-struct CalWorld2
  ([mode : (U 'calendar 'help)]
   [format : CalFormat]
```

```
[calendar-current-date : Date]
[now-date : Date]
[now-date-string : String]
[now-time : Time]))
```

The Time and Date structs ought to be clear enough as written. The components of the CalWorld2 have the following meanings:

- `mode` The project2 calendar application has two modes: the default calendar mode, and a help mode. The user will be able to toggle between the modes with keystrokes.
- `format` This is just as in project1.
- `calendar-current-date` This is the current date indicated on the calendar.
- `now-date` This is the date at which the person sitting in front of the computer is using the application. It doesn't have any connection to calendar-current-date.
- `now-date-string` This is a string equivalent of now-date. Since the date only changes at the rate of once per 24 hours, the string is cached in order to eliminate what would be busy work otherwise. This issue is discussed in more detail below.
- `now-time` The current time of day, also not connected to the calendar.

Note that your project1 code may have to be adapted to work with a `CalWorld2` instead of a `CalWorld`; this is to be expected and is part of the exercise.

The project2 application is completed when it implements the following specifications:

- The run function is of type

```
(: run : CalFormat Integer Integer -> CalWorld2)
```

In addition to a format, run is given a month and a year, and sets the `calendar-current-date` to the first of that month.

- The application continuously updates the current time via `on-tick`. It also advances the date (`now-date`) when the time rolls over to the next day. There are instructions below on how to determine the real-time date and time.
- The date and time now are displayed clearly near the calendar, with the month named, and time, in particular, is shown as am/pm time (not 24-hour time).
- The plus and minus keys move the current day forward and backward, one day at a time, smoothly across month and year boundaries.

- The right and left arrow keys move the current date forward by a month and backward by a month, insofar as that's possible (special cases discussed below*).
- The up and down arrow keys move the current date forward by a year and backward by a year, with the special case that Feb 29 moves forward and back to March 1 of the succeeding/preceding year.
- The capital T key sets the calendar's current date to the actual current date.
- If the application is in calendar mode, the ? key puts it in help mode. In help mode, the escape key puts the application in calendar mode.

*The special cases for moving by months are as follows. If the calendar's current date is advanced into the next month, it's moved either to same day of that month, or, if that is impossible, the highest day possible in the target month. So, for example, advancing from March 31 forward one month goes to April 30 (because there is no April 31). Note that the special-case handling for month movement is a different protocol than the leap-day handling for year movement.

To facilitate moving back and forward one day at a time, implement these two functions:

```
(: yesterday : Date -> Date)
(: tomorrow  : Date -> Date)
```

Not to gush, but if you write these functions thoughtfully, they are simply beautiful. They shouldn't be more than a few lines long each.

Use the following method to get the current date and time. First, up near the top of the file, among the requires, write

```
(require typed/racket/date)
```

This will give you access to, among other things, a function called

```
current-date : -> date*
```

This function consumes no arguments and returns a compound value called a `date*` (with a lowercase d, and the * is part of the name, for reasons unknown to me). The `date*` contains a bundle of information about the moment the function is called. For example, if I run it right now, the result is

```
> (current-date)
```

```
- : date
(date* 42 2 22 26 2 2021 5 56 #f -21600 456152200 "CST")
```

You can read the Racket docs for what all these components mean. In my implementation, I used the following selector functions to read what I needed from the date* result:

```
date-month  : date* -> Integer
date-day    : date* -> Integer
date-year   : date* -> Integer

date-hour   : date* -> Integer
date-minute : date* -> Integer
date-second : date* -> Integer
```

Write the following functions to cobble together the current date and time at any moment:

```
(: read-date-now : -> Date)
(: read-time-now : -> Time)
```

To enable updating the date and time, I wrote a function

```
(: tick : CalWorld2 -> CalWorld2)
```

and attached it to big-bang with

```
[on-tick tick 1/4]
```

This means I am calling the tick function 4 times per second (which might be overkill, but it works). The tick function updates the current time (`now-time`) in the CalWorld2 struct, and if the current time rolls over from one day to the next, it also updates `now-date` and `now-date-string` . It leaves those fields alone most of the time, as they need to be changed relatively rarely.

To summarize, in calendar mode, project2 must

- display the current date clearly in words (see the Panopto demo),
- display the current date with a visual indicator in the corresponding cell in the calendar grid,
- display the now date and now time somewhere alongside the calendar.

You have latitude on the details outside these minimal specifications.

Note that Memorial Day, Labor Day, and Thanksgiving are not officially part of this project. You may continue to indicate them as in project1 (not for extra credit); we've left them out of this project so as to reduce the complexity of the exercise and keep it feasible.

In help mode, the application must display clear information about the available keyboard commands, and return to calendar mode when the escape key is pressed (it will match to the string `"escape"` in the key handling function).

A note on `check-expect` testing. Certain functions like `read-date-now` are nearly impossible to test, and functions that produce complicated images and run worlds are not testable in any remotely easy way. Having said that, there are certain functions (`yesterday` and `tomorrow`, for example) that can be easily tested and should be. There are other functions you might write that seem not to warrant testing -- for example, a function that returns "January" for 1, "February" for 2, etc. Use your judgment on this. We will lighten our usual policy and not insist that you obey a blanket directive to test absolutely everything. You need not. But do make sure your code works as best you can.

Make sure your application works with differently sized formats, and submit it in your repository under `project2/project2.rkt`.