**Term project**: "Tiempo"

**Description**: A scheduling and time-management app. Allows you to create, modify, and automatically "interleave" your calendars via a sleek UI. Has full integration with Google Calendar/iCloud Calendar/Outlook Calendar meaning you can import files from these services, modify them, and export them back as you wish. Also includes a unique task-management system that makes tasks equally as present on your calendar as events (unlike other apps) and which allows you to start immediately working with a built-in pomodoro timer.

**Competitive analysis**: There are a few categories of "competitors" that I decided to look at. To start, for 15-112 projects related to calendaring, there were only two that I could find: Jiayi Wang's Weekly Scheduling and Rachel Luo's Intentional Time. Both had a lot of the same functionalities regarding event manipulation and Rachel's did something similar to my idea of interleaving (albeit slightly less complicated since her program moves around 0 second time intervals which are supposed to be prayer periods, whereas mine will be dealing with larger time intervals). Nonetheless, neither attempted to deal with ics files and instead created their own system independent of ICS. Also, neither one incorporated tasks directly into the calendar view (or did a pomodoro timer) which is something I'm trying to do to differentiate my calendar app.

**Structural plan**: File organization will be as follows:

- main.py -- stores main() and imports functions from files below
- ics_parser.py -- contains functions to take in a path to an ics file, parses the file (harder than it sounds), and returns a "week" or "month" data object for use by calendar functions
- calendar_graphics.py -- contains MVC functions that display data in "week" or "month" objects (OOP) and manipulates that data in the model according to events in the controller (e.g., dragging an event across days)
    - interleaver.py -- called by controller function in calendar_graphics.py. Takes in immutable event choices, wake up and bed times, and finally "week" or "month" objects and converts them to real number intervals that represent free time throughout the week. It then takes the mutable events, cuts them up into 1 hour intervals, and recursively (algorithmic complexity) finds a way to fit them into your calendar such that different mutable events alternate (note this only works with >1 mutable event).
- ics_converter.py -- takes our new calendar with all of its changes and edits the original ics file to reflect the modifications (then somehow allows you to export it… perhaps just manually through file system)
- task_graphics.py -- contains MVC functions for tasks
- pomodoro_graphics.py -- (different mode) contains MVC functions for pomodoro timer

**Algorithmic plan**:

- interleaving.py
    - the goal is to **take in** *event* objects from each day in the month/week which store information about *start time* & *end time* (as datetime objects), as well as

information about their *mutability*, *break lengths,* and *off-limit* times (e.g., sleep) throughout the month/week
    - and then to **generate** a union of real number intervals for each day which represent times that are free of immutable events
    - and then to **loop** over $x_0$ which is the number of mutable events
        - each loop the program cuts up each mutable event into $x_0$ equal-time-length pieces (+ (5 – ___mod(5))
        - and then **tries** to fit them into the time intervals legally such that the pieces alternate in event name (e.g., psychology studying then math studying then english studying then psychology studying etc.)
        - if that size doesn't work then you try $x_0$ + 1 (or decrement the *break lengths*)

**Timeline**:

- [ ] Friday, November 19 – create dictionaries for entire year, complete weekly view, start on interval generation for interleaver
- [ ] Monday, November 22 – create task view integrate with calendar graphics, finish interleaver
- [ ] Tuesday, November 23 – MVP… afterwards continue work on pomodoro timer
- [ ] Friday, November 26 – complete project… get feedback to improve UI