1. Consder supporting g,j,s in Julia constructor and/or 0,1,2 in C constructor, bypassing lastJulianDate specification
2. Support 10/15/2015, 15.10.2015 and 10-Feb-2015 in Julia constructors
3. Consider adding an offset in hours plus fractions of an hour to a lastJulianDate to account for the fact that the transition from Julian to Gregorian must have occurred in local time
4. Update CalCoords based on YMDCoords changes, e.g., checkCalCoordsElements
5. Write asStringCalCoords
6. Change functions that have year, month, dayofmonth, calendar in the argument to take a YMDCoords, if appropriate. Also, functions that call those functions should call checkYMDCoords - dayOfTheMonthFromWeekdayRule. Not sure about isLeapSecondDay, as it should not be used with localDatetime, since leap second does not occur at the end of the day. Perhaps it should take a YMDCoords and check that calendar == 0. Also, isSkipped – at least change the type of the year argument to int\_32t.
7. Check for skipped days in checkYMD, checkDate, checkCal and in create functions before converting to Gregorian calendar
8. Check julian to Gregorian conversions that cause borrow and carry
9. Consider change CalCoords structure to be a YMDCoords plus TimeCoords plus other fields
10. Test createYMDCoordsFromDayOfYear – all branches
11. In Julia, take a string for day of week in YMDCoords constructor from weekday rule
12. Check return value from dayOfTheWeek for errors
13. Create TimeCoords
14. Handle the fact that the transition from Julian to Gregorian occurs in wall time as follows:
    1. Only needs to be considered when a translation from wall, standard, or universal to wall, standard or universal results in a move to an invalid day on the calendar
    2. If the direction of the translation is forward, i.e., universal to standard, standard to wall, or universal to wall with a positive offset, or standard to universal, wall to standard, or wall to universal with a negative offset, the date after the last Julian date must be considered valid. We can’t handle this by merely increasing the last Julian date by 1, since that would also make the first Gregorian date invalid. Instead, we need to set a flag by adding 1e9 to the calendar. (Calendar specifications only require 8 digits).
    3. Similarly, when the direction of the translation is backward, the day before the first Gregorian date must be considered valid. In this case, we add 2e9 to the calendar.
    4. We shouldn’t have to change the functions that check the validity of calendars, as these adjustments only occur when a Date or CalCoords are created during a translation.
    5. Find all calls to lastJulian date and adjust as necessary. Also calls to firstGregorianDate.
15. Need to thoroughly test offsetCalCoords around Julian to Gregorian changes