Installations Required to Run Program

* npm install prompt-sync
* npm install moment
* npm install date-holidays

Usage

* Run command in terminal
  + Node payDateCalculator.js

Holidays Supported

* USA

Library Explanations

* Prompt-sync
  1. Used to get user input
* Moment
  1. Used to get dates
     + Months
     + Days
     + Years
     + Day of the week
     + Adding & Subtracting days/months/years from each other to get accurate days
       - Accounting for months that has 28 days, 30 days, 31 days
* Date-holidays
  1. To ensure I got all of the holidays associated with the USA, some of them I may have been unaware of since some holidays are not main stream holidays

Test Cases

* Special Cases
  1. Holiday being on a Monday
     + Solution
       - Adding a day instead of Subtracting a day
     + Explanation
       - Prevents an infinite loop that conflicts with the weekend condition statement for adding. I didn’t have an explicit instruction on what to do in this situation.
       - Therefore, I added days continuously until there wasn’t a Holiday moving forward
     + Example
       - December 24th – Monday (Christmas Eve)
       - December 25th – Tuesday (Christmas)
       - Solution – 26th of December
* Case
  1. dueDate >= fundDay + 10 & paySpan = weekly
     + Note: paySpan
       - I assumed the initial due date would be 7 days after the fund date due to paySpan being weekly
         * paySpan = ‘bi-weekly’

dueDate = 14 days after fund date

* + - * + paySpan = ‘monthly’

dueDate = next month after fund date

* + - Solution
      * Going to the next paySpan days
        + If paySpan is weekly, 7
        + If paySpan is bi-weekly, 14
        + monthly, next month
    - Explanation
      * If the day does not land on a holiday or weekend, and it does have direct deposit checked as no, then this can become a possible infinite loop
      * Although, I believe it is supposed to be updated each iteration if all conditions are cleared except
        + dueDate >= fundDate + 10
  1. Other cases were made to test the robustness of the code, but the ones explained above I believe were most important to note.
  2. Another test case
     + Ensuring user inputs are valid
       - paySpan
       - fundDay
       - directDeposit