Candidate Test: Workday Calendar

1 Introduction

Welcome to our coding challenge, designed to assess your ability to apply practical software development skills to solve real-world problems. This challenge is open to candidates of all levels, aiming to evaluate your technical proficiency, problem-solving capabilities, and attention to detail through the creation of a Workday Calendar application for a Supply Chain Management system.

2 Use Case: Supply Chain Management

Envision a scenario where a manufacturing entity needs to schedule the production of an item dependent on components from various suppliers. Given the known lead times for these components, measured in working days, production cannot start until all parts are received. Your Workday Calendar will assist in calculating:

The critical order date and time for each component to align with the production schedule.

The anticipated delivery dates for the final products to customers, factoring in production and shipping durations.

3 Challenge Description

Your objective is to build a Workday Calendar application that accurately calculates the date and time resulting from the addition or subtraction of a specified number of working days to/from a given date.

Application Interface

User-Friendly Design: The application may feature a simple Command Line Interface (CLI) or Graphical User Interface (GUI), but it should prioritize ease of use, guiding users smoothly through setting holidays, defining working hours, and calculating dates.

Core Functional Requirements

Working Days:

Definition: Automatically defined as Monday through Friday, excluding weekends.

Holiday Exclusions: Capable of excluding designated holidays from the count of working days for accurate calculations.

Holidays:

Customization: Users must have the ability to designate specific dates as holidays.

Types of Holidays: The system should intelligently differentiate between fixed holidays (occurring on a specific date) and recurring annual holidays (e.g., observed annually on the same day), offering comprehensive holiday management.

Working Hours:

Precision in Calculations: The application extends its functionality to incorporate working hours, ensuring calculations account for the exact time work starts or ends. This feature is crucial for operations involving partial working days, enabling the application to offer detailed insights into the calculation of starting or ending times within the working day framework.

3.1 Examples

1. Basic Working Day Calculation

Scenario: Calculate the result of adding a quarter of a working day to a given start datetime.

Input: Start datetime = 24.05.2004 15:07, Add 0.25 working days, Workday start = 08:00, end = 16:00

Expected Output: The resulting datetime should be the next day at 09:07, assuming the next day is a working day and not a holiday.

2. Midnight Boundary and Fractional Working Day

Scenario: Calculate the result of adding half a working day to a start datetime that begins before the working hours.

Input: Start datetime = 24.05.2004 04:00, Add 0.5 working days, Workday start = 08:00, end = 16:00

Expected Output: The resulting datetime should be the same day at 12:00 (noon), as half a working day from the start of working hours (08:00) leads to 12:00.

3. Working Day Calculation with Negative Days

Scenario: Calculate the result of subtracting 5.5 working days from a given start datetime.

Input: Start datetime = 24.05.2004 18:05, Subtract 5.5 working days, Set recurring holiday = 17 May, Set single holiday = 27 May 2004, Workday start = 08:00, end = 16:00

Expected Output: The resulting datetime should be 14.05.2004 12:00. This calculation must account for the non-working days (weekends) and the specified holidays.

Additional Correct Results:

These examples demonstrate how the system should handle various additions of working days to specific start datetimes, considering the start and end of the working day, weekends, and holidays. Set recurring holiday = 17 May, Set single holiday = 27 May 2004, Workday start = 08:00, end = 16:00.

- **Example 1:**
  - Input: 24-05-2004 19:03, Add 44.723656 working days,
  - Expected Output: 27-07-2004 13:47
- **Example 2:**
  - Input: 24-05-2004 18:03, Subtract 6.7470217 working days
  - Expected Output: 13-05-2004 10:01
- **Example 3:**
  - Input: 24-05-2004 08:03, Add 12.782709 working days
  - Expected Output: 10-06-2004 14:18
- **Example 4:**
  - Input: 24-05-2004 07:03, Add 8.276628 working days
  - Expected Output: 04-06-2004 10:12

4 Evaluation Criteria

Your submission will be evaluated based on:

Functionality: Accuracy of the date calculations, handling of edge cases, and adherence to the defined requirements.

Testability and Tests: The ease with which your code can be tested, and test coverage and thoroughness.

Code Quality: Readability, use of design patterns, and best coding practices (SOLID, Clean Code, DRY, KISS, etc.) where appropriate.

5 Submission Instructions

Please submit your solution preferably as a link to a GitHub repository, but alternative submission methods are also acceptable if necessary.

We're excited to see your approach to solving this challenge and how you apply your coding skills to create a practical and efficient application.

Good luck!