CARTER BALLOW REPORT\_PROJ\_1

806389359

### Project Status & Summary

All of the parts of the project have been implemented according to the specification provided to me to the best of my ability. This includes the BloodDonation, DateUtility, and VacationAccount classes and the main file for testing.

**ISSUES I HAD - Date Logic Consolidation:** Initially, date validation logic was scattered. The BloodDonation::setDate had another method in the DloodDonation class which what a bad use of memory so I fully implemented the isValidDate helper into the code for the setter. His was the biggest issue I had, and then also dealing with the logic of the valid date and making sure to check if the date setter didn’t wrok in the Vacation Account.

**Test Case Documentation**

**1. DateUtility Function Tests:**

* **Purpose:** To directly make sure that t the core logic was correct, I had to first test the static helper functions before they are used by other classes.
* **Tests:**
  + isValidDate: Checks various date formats, including boundary years (1900, 2100), incorrect lengths, invalid separators, and non-numeric characters. It also confirms that days up to 31 are accepted for any month and all of these passed so I was happy about this.
  + isSameDay: Confirms that two identical, valid date strings return true and different ones return false. It makes sure to do the opposite if one is not in the right format using the is Valid Dates.
  + isSixMonthsApart: Tests scenarios for exactly 6 months, more than 6 months, less than 6 months, and date comparisons that cross a year boundary. It also verifies that the order of dates does not matter and that it returns false if an invalid date is used. Befor ei had a hard time doing the fact that it could also be before the input date.

**2. BloodDonation Class Validation Tests:**

* **Purpose:** To ensure the setters in the BloodDonation class correctly validate input and set member variables to their default invalid state (-1 or "") upon failure, I implemented these tests to keep track of anything that is wrong before the vacation account ran.
* **Tests:** Verifies that valid data is accepted correctly. It then tests edge cases for ID (too short), Age (too young), Weight (too light), and Date (invalid format) to confirm they are properly rejected.

**3. VacationAccount Scenario Tests:** Five distinct VacationAccount objects were created to test different operational scenarios.

* **Account 1:**
  + **Purpose:** To simulate a standard history function I just made sure there was an account with good proficiency.
* **Account 2:**
  + **Purpose:** To test the donation rejection logic. This case verifies that the system correctly rejects donations made on the **same day** as a previous one and donations made **less than six months** later. All of the tests passed for this.
* **Account 3:** 
  + **Purpose:** To test the system's response invalid data passed to addVacationToAccount.
  + **Tests:** This includes attempts to add donations with an invalid ID, Age, Weight, or Date format, as well as passing a nullptr. It confirms that all these attempts fail and the balance remains zero until a fully valid donation is given too the system
* **Account 4:**
  + **Purpose:** To ensure that a VacationAccount only exceptw donations where the BloodDonation ID matches the account's own ID. It verifies that donations for other employees are rejected.
* **Account 5 and or 6**
  + **Purpose:** To verify that the VacationAccount copy constructor performs a deep copy and that the original object and its copy operate independently after creation.
  + **Tests:** An original account is created and populated. A copy is made. The test then adds a new donation to the original account and asserts that only the original's balance changes. This I don’t think even needed to be used cause Idk why the function was there It then adds a donation to the copied account and asserts that only the copy's balance changes, proving there are no shared memory issues.
* These are not all the tests just the ones I focused on writing.