Group Member Names:

Vincent Lee, Gustavo Necochea Aguayo, Emmett Lim, Andrew Arsenault

Algorithm 3 Pseudocode:

Problem Statement:

Determine the available time slots for group meetings based on individual schedules and daily working periods.

Input:

Busy_Schedules: A list of schedules where each schedule contains unavailable time intervals for a person
Working_Periods: A list of the earliest and latest available times for each group member
Duration: Minimum required meeting duration

Output:

Available Slots: A list of time intervals when all group members are available

Assumptions:

Times (24-hour military) -> change to minutes

```
function schedule(person1_Schedule, person1_DailyAct, person2_Schedule, person2_DailyAct,
duration_of_meeting):

1. Initialize empty arrays to store converted schedules:
        Create two arrays: person1Array and person2Array, to store the converted busy intervals
        (in minutes) for both people

2. Convert Person1 and Person2 schedules to minutes:

3. Combine Busy Schedules:
        Combine both person1Array and person2Array into a single list of busy intervals.
        Sort the combined list by the start time of each interval.

4. Find Available Time Slots:
        Identify gaps between the busy intervals where a meeting can be scheduled.
        Ensure each gap is large enough for the duration of the meeting.

5. Filter by Working Periods:
        Check that the available meeting times fall within the shared working periods
        person1_DailyAct and person2_DailyAct.

6. Output Available Slots:
```

Proving Efficiency for Pseudocode:

Step Count:

1. Convert Time to Minutes: O(n)
   - This step converts each schedule entry from hours to minutes.

2. Store Time: O(n)
   - Each converted time is stored in a new array.

3. Combine Schedules: O(n1+n2)
   - where n1 is the number of entries in person 1's schedule and n2 is the number of entries in person 2's schedule.
   - This step merges the schedules of both people into one combined list.

4. Sort Combined Schedules: O(nlogn)
   - Sorting the combined list of schedules: most time consuming operation, since sorting an array of size n takes O(nlogn)

5. Find Gaps: O(n)
   - After sorting, finding gaps between the busy periods to identify available meeting times.

Since sorting is the longest task in this process
Time Complexity: O(nlogn)