The main driving function in the project will be name **greedy**. In the function, we are passed 5 parameters, **input1, act1, input2, act2,** and **duration. Input1 and input2** represents the persons 1 and 2 daily activitys when they are not free. **Act1** and **act2** represents the times when person1 and person2 are available to meet. **Duration** represents the minimum meeting time.

We start the function off with replacing all times from the inputs to floats in order to perform arithmetic and comparisons. We then find the agreed interval by getting the maximum start and minimum end from **act1** and **act2.** This agreed interval will be under the variable **interval** which is a array of size 2. We then checked for the edge case where there is no agreed time slots for both persons to meet and no inputs for both persons.

Passing this edge case, we then join both **input1** and **input2** and sort the array in ascending order using the end time as the sort condition. This new array will be named **joined**. Joining the 2 arrays will take **O(m + n)** time and take up **O(n + m)** memory space where **m** and **n** is based on the length of **input1** and **input2**. Sorting the joined array will have a **O(nlogn)** time complexity.

The main algorithm after obtaining the sorted array would be checking if there is a minimum valid meeting duration from out absolute start time from **interval** to the 0th index of **joined.** We then enter a while loop checking for overlaps. We can check this by if the current end time is greater than the next start time. If it is, we would calculate the max end time starting at the max of (curr_start, next_start), if there is a valid meeting time, increment pointer by 2. If there isnt, we would want to check if there is a valid minimum duration of **duration** minutes to the next start, increment pointer by 1. Lastly, we would want to check the last end time from the **joined** list to the absolute end time in **interval**. This whole process would take **O(n)** time where n is dependent on the length of **joined**.

I also have a helper function to calculate arithmetic for times. “**timeAddition**” adds the current time by the duration once. **“getMaxEnd”** recursivly calls **“timeAddition”** with a base case if duration added to the current time is greater than its bounds.