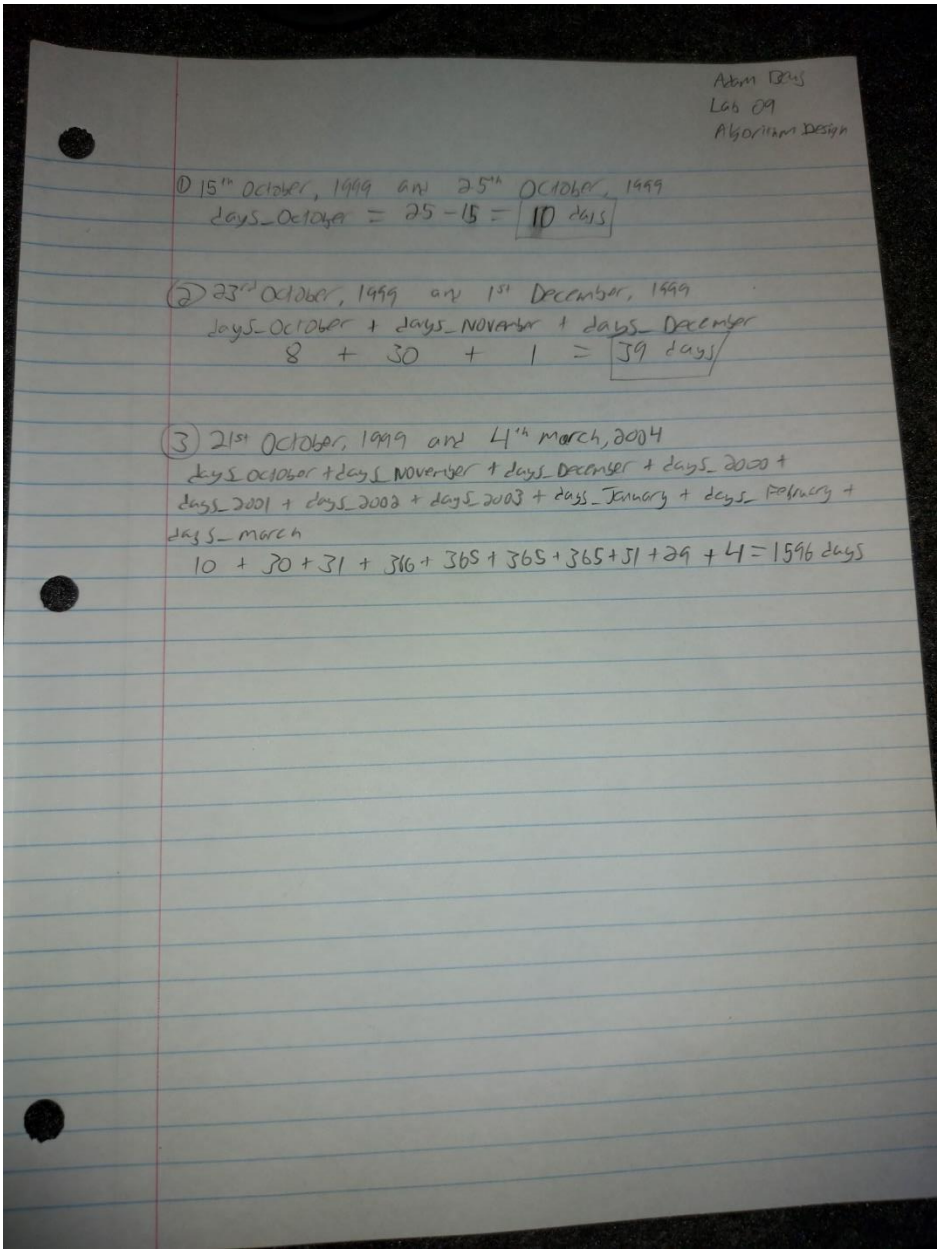
Step 1: By Hand



Step 2: Approach

To solve this problem, I will first check to see if the two dates fall within the same year, then the same month. If they fall within the same month, I will simply calculate the

end date minus the start date, and that is my answer. If it is the same year but not the same month, I will calculate the number of days left in the month of the start date, and calculate the rest of the time keeping in mind the number of days in each month. If the two dates are in different years, I will calculate the number of days left in the start date's year, and the days passed so far in the end year. For the years in between, I will add either 365 or 366 days, depending on whether or not it is a leap year.

Step 3: Pseudocode

```
days_in_months = [31, 28 + is_leap_year(start_year), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
if end_year = start_year
        if end_month = start_month:
                total_days = end_day – start_day
        else:
                for month in range(start_month + 1, end_month - 1)
                        total_days += days_in_month[month]
                total_days += end_day
else:
        for month in range(start_month + 1, end_month - 1)
                total_days += days_in_month[month]
        total_days += end_day
        for year in range(start_year + 1, end_year – 1)
                if is_leap_year:
                        total_days +=366
                else:
                        total_days += 365
```

```
        total_days += end_day
```

Step 4: AI

```
Function daysInMonth(month, year):
  if month == 1 or month == 3 or month == 5 or month == 7 or month == 8 or month == 10 or month == 12:
    return 31
  else if month == 4 or month == 6 or month == 9 or month == 11:
    return 30
  else if month == 2:
    if isLeapYear(year):
      return 29
    else:
      return 28

Function daysBetweenDates(startDate, endDate):
  startYear = startDate.year
  startMonth = startDate.month
  startDay = startDate.day
  endYear = endDate.year
  endMonth = endDate.month
  endDay = endDate.day

  totalDays = 0

  if startYear == endYear:
    if startMonth == endMonth:
      totalDays = endDay - startDay
```

```
    else:
        totalDays += daysInMonth(startMonth, startYear) - startDay
        for month in range(startMonth + 1, endMonth):
            totalDays += daysInMonth(month, startYear)
        totalDays += endDay
  else:
    totalDays += daysInMonth(startMonth, startYear) - startDay
    for month in range(startMonth + 1, 13):
        totalDays += daysInMonth(month, startYear)

    for year in range(startYear + 1, endYear):
        if isLeapYear(year):
            totalDays += 366
        else:
            totalDays += 365

    for month in range(1, endMonth):
        totalDays += daysInMonth(month, endYear)
    totalDays += endDay

  return totalDays
```

Step 5. Compare and Contrast

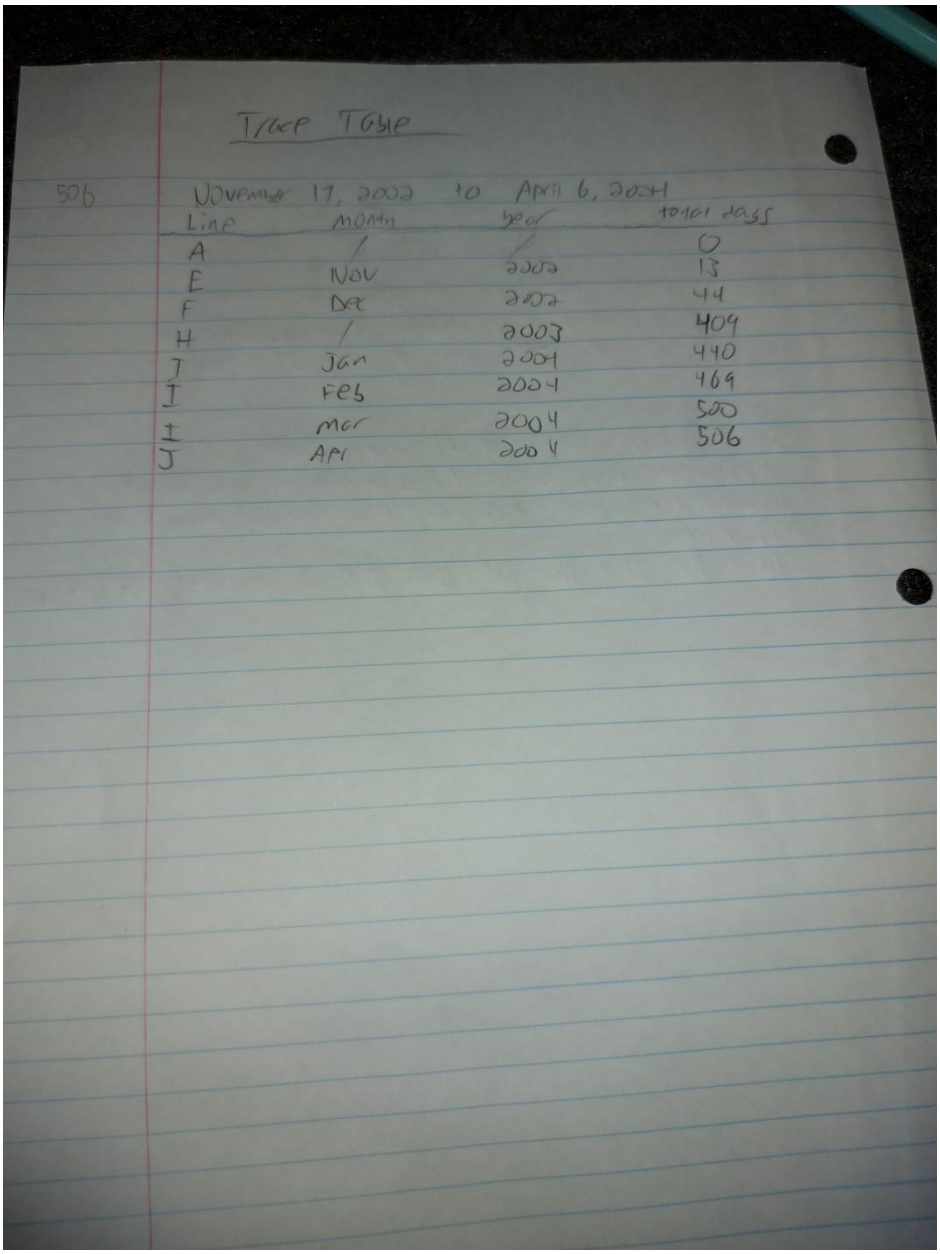
My pseudocode is much shorter and more simple, but may not be as effective. It is quite smart to do what the AI did and turn the days in month into a function that can be called. However, I preferred my approach in turning the number of days in each month into

a list. One thing my code was missing was initializing total_days to 0 at the beginning. Both versions of this pseudocode are effective and follow the algorithm described in step 2.

Step 6: Updated Pseudocode

```
days_in_months = [31, 28 + is_leap_year(start_year), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
total_days = 0                                                                    (A
if end_year = start_year
    if end_month = start_month:
        total_days = end_day – start_day                                          (B
    else:
        total_days += days_in_months[start_month- 1] - start_day
    (C
        for month in range(start_month + 1, end_month - 1)
            total_days += days_in_month[month]                                    (D
        total_days += end_day
else:
    total_days += days_in_months[start_month- 1] - start_day                      (E
    for month in range(start_month + 1, end_month - 1)
        total_days += days_in_month[month]                                        (F
    for year in range(start_year + 1, end_year – 1)
        if is_leap_year:
            total_days +=366                                                      (G
        else:
            total_days += 365                                                     (H
    for month in range(0, end_month – 1)
        total_days += days_in_month[month]                                        (I
    total_days += end_day                                                         (J
```

Step 7: Trace



Step 8: Efficiency

This algorithm is of O(n) efficiency, because it is dependent on the number of years. The other calculations, including checking to see if the years are the same and cxalculating the days left in the year, are O(1) because they run up to a fixed number of times.