Lab 05

Step 1: By Hand

Time Taken: about 10 minutes

We need to find a number in the list. We will use the following list:

```
[11, 13, 28, 32, 36, 39, 41, 48, 49, 52, 54, 57, 64, 68, 75, 79,
80, 82, 91, 92]
```

We will try to find the number `32`, which is in the list.

```
[11, 13, 28, 32, 36, 39, 41, 48, 49]
[11, 13, 28, 32]
[28, 32]
[32]
```

Next, we need to find a number *not* in the list. We will use the following list:

```
[3, 13, 16, 17, 19, 25, 30, 33, 36, 37, 46, 56, 57, 58, 59, 64,
73, 81, 95, 99]
```

We will try to find the number `96` which is *not* in the list.

```
[46, 56, 57, 58, 59, 64, 73, 81, 95, 99]
[64, 73, 81, 95, 99]
[95, 99]
```

```
[99]
[]
```

Step 2: Approach

Time Taken: about 10 minutes

We start with a sorted list of random numbers, and a target number. Find the greatest index and the lowest index of the list. We need to start by taking the highest index, subtract the lowest index, and divide by 2 rounding down. Using that number as the index of the list, we check if the number is greater than, less than, or equal to the target number. If it is equal, we are done. If it is less than, take the index of the number below and make that the greatest number we can check. If it is greater than, take the index of the number right above and make that the greatest index we can check. If the index of the max number we can check and the lowest number we can check are the same, and the number at that index isn't the target number, then the number was not found and we will inform the user of that. Repeat the steps after getting the highest and lowest index until the target number is either found, or is discovered to not be in the list.

Step 3: Pseudocode

Time Taken: about 15 minutes

Using the description I wrote in the previous step, I made the following pseudocode.

```
numbers = list of sorted random numbers
targetNumber = target number
maxIndex = highest index
minIndex = 0

WHILE True
```

```
    indexDifference = maxIndex - minIndex

    IF indexDifference == 0
        IF minIndex == targetNumber
            RETURN minIndex
        ELSE
            RETURN not found

    checkIndex = (minIndex + maxIndex) / 2

    IF checkIndex == targetNumber
        RETURN checkIndex

    IF targetNumber < checkIndex
        maxIndex = checkIndex - 1
    ELSE IF targetNumber > checkIndex
        minIndex = checkIndex + 1
```

Step 4: Gemini

Time Taken: about 3 minutes

The instructions asked for Copilot, but I felt like asking Gemini instead. Using my paragraph from Step 2, I asked Gemini to create pseudocode for the algorithm. Here is the generated pseudocode:

```
function searchSortedList(list, target):
  lowIndex = 0  ; Initialize lowest index
  highIndex = length(list) - 1  ; Initialize highest index

  while lowIndex <= highIndex:
    midIndex = floor((highIndex - lowIndex) / 2) + lowIndex  ;
Calculate middle index (rounded down)
```

```
    if list[midIndex] == target:
      return midIndex  ; Target found at middle index
    elif list[midIndex] < target:
      lowIndex = midIndex + 1  ; Update lowest index
    else:
      highIndex = midIndex - 1  ; Update highest index

  return "Target not found"  ; Target not found in the list
```

Step 5: Compare and Contrast

Time Taken: about 15 minutes

My solution provides a way to check that there's only one number, which allows you to then check one last time if the number is found or not. Gemini's solution does not do this. My solution also provides a quicker way to find the middle index. The AI solution is more to the point.

My solution can be improved based on the AI solution by shortening the pseudocode. This is because the AI wrote it in better pseudocode, despite being mostly the same opperations. The AI also made it so the code returns "not found" simply by putting the highIndex below the lowIndex. This is an easy way to check if it's found while reducing the amount of logic.

Gemini's solution could be improved by decreasing the number of calculations needed to find the middle index. My way to find the middle index is quicker than the AI's way.

Both solutions accurately match the algorithm I performed in Part 1. They do it slightly differently, however.

Step 6: Update

Time Taken: about 5 minutes

The following is my updated pseudocode:

```
search(numbers, targeNumber)
    maxIndex = numbers.length() - 1
    minIndex = 0

    WHILE maxIndex >= minIndex
        checkIndex = (minIndex + maxIndex) / 2

        IF numbers[checkIndex] == targetNumber
            RETURN checkIndex
        ELSE IF numbers[targetNumber] < checkIndex
            maxIndex = checkIndex - 1
        ELSE IF numbers[targetNumber] > checkIndex
            minIndex = checkIndex + 1
    RETURN "not found"
```