Step 1: By Hand

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

Looking for 8:

1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21

Looking for 13:

1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21
1	2	3	4	5	6	7	8	9	10	11	12	14	15	16	17	18	19	20	21

Not found

Step 2: Approach

First get the desired value from the user to be searched for in the list. Find the average of the first and last index in the list. Check if the value at that index is the desired value, if it is then put that on the screen for the user to know the value was found. If not, check if the first index and last index equal each other because if they do then you've gone through the whole list and should put that the desired value wasn't found. Then check if the value at the average index comes before or after the desired value in this sorted list (less than or greater than for numbers). If the desired value comes before (is less than) the value found at the average index, use the index before the current average index and find the average index between that one and the first index to repeat the process. If the desired value comes after (is greater than) the value found at the average index, use the index after the current average index and find the average index between that one and the last index to repeat the process. Continue this process until the desired value is found or until you've run out of indexes to check because the first and last is the same index and it's not the desired value.

Step 3: Pseudocode

```
PROMPT for desiredValue to be searched for in the list

GET firstIndex in list
GET lastIndex in list

notFound ← true
WHILE notFound
      averageIndex ← (firstIndex + lastIndex) / 2

      IF averageIndex = desiredValue
            PUT that the desired value was found in the list
            notFound ← false

      IF firstIndex = lastIndex
            PUT that the desired value was not found in the list
            notFound ← false

      IF averageIndex < desiredValue
            lastIndex ← averageIndex – 1

      IF averageIndex > desiredValue
            firstIndex ← averageIndex + 1
```

Step 4: Copilot

Plaintext Copy

```
1. Get the desired value (target) from the user.
2. Initialize first index (low) to 0.
3. Initialize last index (high) to the length of the list minus 1.
4. While low is less than or equal to high:
    a. Calculate the middle index (mid) as the average of low and high.
    b. If the value at mid equals target:
        i. Print "Value found at index mid".
        ii. Exit.
    c. Else, if low equals high and value at mid is not target:
        i. Print "Value not found".
        ii. Exit.
    d. Else, if value at mid is less than target:
        i. Set low to mid + 1.
    e. Else, if value at mid is greater than target:
        i. Set high to mid - 1.
5. If loop exits, print "Value not found".
```

Step 5: Compare and Contrast

- Provide an analysis as to the pros and cons of the two solutions?
  - The pros of the copilot solution is it doesn't use an extra variable to go through the loop. Cons is that it uses short circuiting which isn't bad in this case but it's bad practice. Pros of my solution is I follow the pseudocode rules better. Cons is I made a mistake in just comparing the average index instead of the value at that index.

- How can your solution be improved based on what Copilot provided?
  - I can make the comparisons by accessing the value at the average index rather than comparing the index itself.

- How can Copilot's solution be improved based on what you know?
  - It can be improved by incorporating the pseudo code rules to make it easier to understand and read

- Does the pseudocode in Step 3 and Step 4 match the algorithm you performed in Step 1?
  - Yes it does match pretty closely.

Step 6: Update

```
PROMPT for desiredValue to be searched for in the list

GET firstIndex in list
GET lastIndex in list

notFound ← true
WHILE notFound
      averageIndex ← (firstIndex + lastIndex) / 2

      IF list[averageIndex] = desiredValue
            PUT that the desired value was found in the list
            notFound ← false

      IF firstIndex = lastIndex
            PUT that the desired value was not found in the list
            notFound ← false

      IF list[averageIndex] < desiredValue
            lastIndex ← averageIndex – 1

      IF list[averageIndex] > desiredValue
            firstIndex ← averageIndex + 1
```