My list:
[10, 7, 3, 2, 6, 4, 1, 8, 9, 5]
By hand:
[5, 7, 3, 2, 6, 4, 1, 8, 9, 10]
[5, 7, 3, 2, 6, 4, 1, 8, 9, 10]
[5, 1, 3, 2, 6, 4, 7, 8, 9, 10]
[5, 1, 3, 2, 4, 6, 7, 8, 9, 10]
[4, 1, 3, 2, 5, 6, 7, 8, 9, 10]
[2, 1, 3, 4, 5, 6, 7, 8, 9, 10]
[2, 1, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

For each item in the list, the code will check if it is greater than what's in the spot highest in the list. When that spot gets filled, it will have the highest value so the code will exclude it from future searches. This will run until the list is completely sorted.

My pseudocode:

```
START
for i in range 1-10:
   for j in range 10-i-1:
      if j > j+1:
         switch j and j+1
   return sorted list
```

CoPilot's pseudocode:

```
def sort_list(input_list):
   sorted_list = []
   while input_list:
      highest = input_list[0]
      for item in input_list:
         if item > highest:
            highest = item
      sorted_list.append(highest)
      input_list.remove(highest)
   return sorted_list
```

Analysis:
Microsoft CoPilot's code is certainly different and saves the sorted list into its own list. This can be useful if you want to preserve the original list for any reason, but in this case is fairly useless. One thing it did that I didn't was define the function; I just jumped straight into the code when it's better to define a function first so it can be reused later. My code from Step 3 actually matches my algorithm by moving

from item to item while AI's list simply adds the largest to another group rather than switching it in the same group.

My updated pseudocode:

```
define sort_list(list)
  for i in range 1-10:
      for j in range 10-i-1:
         if j > j+1:
            switch j and j+1
      return list
```

Trace:

```
1: def sort_list(list):
2:     list_length = len(list)
3:     for i in range(list_length-1):
4:         for j in range(list_length-i-1):
5:             if list[j] > list[j+1]:
6:                 list[j], list[j+1] = list[j+1], list[j]
7:     return list
```

Line	i	j	lst after operation
2	-	-	[26, 6, 90, 55]
3	0	-	[26, 6, 90, 55]
4	0	0	[6, 26, 90, 55]
4	0	1	[6, 26, 90, 55]
4	0	2	[6, 26, 55, 90]
3	1	-	[6, 26, 55, 90]
4	1	0	[6, 26, 55, 90]
4	1	1	[6, 26, 55, 90]
3	2	-	[6, 26, 55, 90]
4	2	0	[6, 26, 55, 90]
3	3	-	[6, 26, 55, 90]
7	-	-	[6, 26, 55, 90]

Efficiency:
O(n^2)

Timing:
Step 1 By Hand: 10 minutes
Step 2 Approach: 6 minutes
Step 3 Pseudocode: 30 minutes
Step 4 Copilot: 6 minutes
Step 5 Compare and Contrast: 10 minutes
Step 6 Update: 7 minutes

Step 7 Trace: 20 minutes
Step 8 Efficiency: 10 minutes