Please just execute the “insertion sort mehrdad” and you can see your requested input and output based on your framework.

1 **for** j = 2 **to** A. *length*

2 *key* = A[j]

3 **//** Insert A[j] into the sorted the sequence A[1 …j-1]

4 i = j \_ 1

5 **while** i > 0 and A[i]> *key*

6 A[i + 1]= A[i]

7 i = i \_ 1

8 A[i + 1]= *key*

import random  
import time  
  
# Step 1: Generate input data sets  
def generate\_data(size):  
 data = [random.randint(0, 1000000) for \_ in range(size)]  
 return data  
  
sizes = [15000, 30000, 60000, 120000, 240000]  
categories = ['random', 'sorted', 'reverse\_sorted', 'identical']  
  
for size in sizes:  
 for category in categories:  
 data = []  
 if category == 'random':  
 data = generate\_data(size)  
 elif category == 'sorted':  
 data = sorted(generate\_data(size))  
 elif category == 'reverse\_sorted':  
 data = sorted(generate\_data(size), reverse=True)  
 elif category == 'identical':  
 data = [random.randint(0, 1000000)] \* size  
 with open(f'{category}\_{size}.txt', 'w') as file:  
 for number in data:  
 file.write(f"{number}\n")  
  
# Step 2: Implement insertion sort algorithm  
def insertion\_sort(arr):  
 comparisons = 0  
 start\_time = time.time()  
 for j in range(1, len(arr)):  
 key = arr[j]  
 i = j - 1  
 while i >= 0 and arr[i] > key:  
 comparisons += 1  
 arr[i + 1] = arr[i]  
 i = i - 1  
 arr[i + 1] = key  
 end\_time = time.time()  
 execution\_time = (end\_time - start\_time) \* 1000 # in milliseconds  
 return comparisons, execution\_time  
  
# Step 3 and 4: Analyze and report performance  
for size in sizes:  
 for category in categories:  
 with open(f'{category}\_{size}.txt', 'r') as file:  
 input\_data = [int(line.strip()) for line in file]  
 comparisons, execution\_time = insertion\_sort(input\_data)  
 print(f"{size},{comparisons},{execution\_time}")

Outputs:

15000,55817695,5880.945205688477

15000,0,1.9948482513427734

15000,112492372,11126.844882965088

15000,0,2.2804737091064453

30000,224784932,24418.71929168701

30000,0,2.9985904693603516

30000,449984549,44547.611951828

30000,0,2.99835205078125

60000,896327754,114745.51248550415

60000,0,5.983829498291016

60000,1799968180,219446.1908340454

60000,0,6.992101669311523

120000,3600634011,475505.63621520996

120000,0,12.758493423461914

120000,7199932779,905669.230222702

120000,0,18.949031829833984

240000,14379497437,2391614.8409843445

240000,0,32.9134464263916

240000,28799851220,3951334.481716156

240000,0,35.903215408325195