Started on	Monday, 7 April 2025, 3:39 PM
State	Finished
Completed on	Monday, 7 April 2025, 4:15 PM
Time taken	35 mins 31 secs
Grade	**100.00** out of 100.00

Write a python program to implement quick sort on the given float array values.

For example:

Input	Result
5 6.9 8.3 2.1 1.5 6.4	left: [] right: [] left: [] right: [] left: [1.5] right: [6.4] left: [] right: [] left: [1.5, 2.1, 6.4] right: [8.3] [1.5, 2.1, 6.4, 6.9, 8.3]
6 3.1 2.4 5.6 4.3 6.2 7.8	left: [] right: [] left: [] right: [] left: [] right: [] left: [] right: [7.8] left: [4.3] right: [6.2, 7.8] left: [2.4] right: [4.3, 5.6, 6.2, 7.8] [2.4, 3.1, 4.3, 5.6, 6.2, 7.8]

Answer: (penalty regime: 0 %)

```python
1  def qsort(L):
2      if L==[]:
3          return[]
4      pivot=L[0:1]
5      left=qsort([x for x in L[1:]if x<L[0]])
6      right=qsort([x for x in L[1:]if x>=L[0]])
7      print("left: ",left)
8      print("right: ",right)
9      return left+pivot+right
10 list1=[]
11 n=int(input())
12 for i in range(n):
13     list1.append(float(input()))
14 print(qsort(list1))
```

	Input	Expected	Got	
✔	5 6.9 8.3 2.1 1.5 6.4	left: [] right: [] left: [] right: [] left: [1.5] right: [6.4] left: [] right: [] left: [1.5, 2.1, 6.4] right: [8.3] [1.5, 2.1, 6.4, 6.9, 8.3]	left: [] right: [] left: [] right: [] left: [1.5] right: [6.4] left: [] right: [] left: [1.5, 2.1, 6.4] right: [8.3] [1.5, 2.1, 6.4, 6.9, 8.3]	✔
✔	6 3.1 2.4 5.6 4.3 6.2 7.8	left: [] right: [] left: [] right: [] left: [] right: [] left: [] right: [7.8] left: [4.3] right: [6.2, 7.8] left: [2.4] right: [4.3, 5.6, 6.2, 7.8] [2.4, 3.1, 4.3, 5.6, 6.2, 7.8]	left: [] right: [] left: [] right: [] left: [] right: [] left: [] right: [7.8] left: [4.3] right: [6.2, 7.8] left: [2.4] right: [4.3, 5.6, 6.2, 7.8] [2.4, 3.1, 4.3, 5.6, 6.2, 7.8]	✔
✔	8 1.2 1.3 4.2 5.3 6.4 7.3 6.8 9.2	left: [] right: [] left: [] right: [] left: [6.8] right: [9.2] left: [] right: [6.8, 7.3, 9.2] left: [] right: [6.4, 6.8, 7.3, 9.2] left: [] right: [5.3, 6.4, 6.8, 7.3, 9.2] left: [] right: [4.2, 5.3, 6.4, 6.8, 7.3, 9.2] left: [] right: [1.3, 4.2, 5.3, 6.4, 6.8, 7.3, 9.2] [1.2, 1.3, 4.2, 5.3, 6.4, 6.8, 7.3, 9.2]	left: [] right: [] left: [] right: [] left: [6.8] right: [9.2] left: [] right: [6.8, 7.3, 9.2] left: [] right: [6.4, 6.8, 7.3, 9.2] left: [] right: [5.3, 6.4, 6.8, 7.3, 9.2] left: [] right: [4.2, 5.3, 6.4, 6.8, 7.3, 9.2] left: [] right: [1.3, 4.2, 5.3, 6.4, 6.8, 7.3, 9.2] [1.2, 1.3, 4.2, 5.3, 6.4, 6.8, 7.3, 9.2]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Write a python program to implement binary search on the given list of string values using iterative method

For example:

Test	Input	Result
binarySearchAppr(arr, 0, len(arr)-1, x)	5 one two three four five two	Element is present at index 4
binarySearchAppr(arr, 0, len(arr)-1, x)	6 one three five seven nine eleven thirteen	Element is not present in array

Answer: (penalty regime: 0 %)

```python
def binarySearchAppr(arr,low,high,x):
    for i in range(0,len(arr)):
        if arr[i]==x:
            return i
    return -1
n=int(input())
arr=[]
for i in range(0,n):
    arr.append(input())
x=input()
res=binarySearchAppr(arr,0,len(arr)-1,x)
if(res==-1):
    print("Element is not present in array")
else:
    print("Element is present at index",n-res)
```

	Test	Input	Expected	Got	
✔	binarySearchAppr(arr, 0, len(arr)-1, x)	5 one two three four five two	Element is present at index 4	Element is present at index 4	✔
✔	binarySearchAppr(arr, 0, len(arr)-1, x)	6 one three five seven nine eleven thirteen	Element is not present in array	Element is not present in array	✔

	Test	Input	Expected	Got	
✔	binarySearchAppr(arr, 0, len(arr)-1, x)	4 two four six eight six	Element is present at index 2	Element is present at index 2	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Write a python program to implement merge sort using iterative approach on the given list of float values.

For example:

Test	Input	Result
Merge_Sort(S)	5 10.2 21.3 3.5 7.8 9.8	The Original array is: [10.2, 21.3, 3.5, 7.8, 9.8] Array after sorting is: [3.5, 7.8, 9.8, 10.2, 21.3]
Merge_Sort(S)	6 20.3 41.2 5.3 6.2 8.1 65.2	The Original array is: [20.3, 41.2, 5.3, 6.2, 8.1, 65.2] Array after sorting is: [5.3, 6.2, 8.1, 20.3, 41.2, 65.2]

Answer: (penalty regime: 0 %)

```python
1  def Merge_Sort(S):
2      if len(S) < 2:
3          return S
4
5      mid = len(S) // 2
6      y = Merge_Sort(S[:mid])
7      z = Merge_Sort(S[mid:])
8      result = []
9
10     i = 0
11     j= 0
12
13     while i < len(y) and j < len(z):
14         if y[i] > z[j]:
15             result.append(z[j])
16             j+=1
17         else:
18             result.append(y[i])
19             i+=1
20     result += y[i:]
21     result += z[j:]
22     return result
```

	Test	Input	Expected	Got	
✔	Merge_Sort(S)	5 10.2 21.3 3.5 7.8 9.8	The Original array is: [10.2, 21.3, 3.5, 7.8, 9.8] Array after sorting is: [3.5, 7.8, 9.8, 10.2, 21.3]	The Original array is: [10.2, 21.3, 3.5, 7.8, 9.8] Array after sorting is: [3.5, 7.8, 9.8, 10.2, 21.3]	✔
✔	Merge_Sort(S)	6 20.3 41.2 5.3 6.2 8.1 65.2	The Original array is: [20.3, 41.2, 5.3, 6.2, 8.1, 65.2] Array after sorting is: [5.3, 6.2, 8.1, 20.3, 41.2, 65.2]	The Original array is: [20.3, 41.2, 5.3, 6.2, 8.1, 65.2] Array after sorting is: [5.3, 6.2, 8.1, 20.3, 41.2, 65.2]	✔
✔	Merge_Sort(S)	4 2.3 6.1 4.5 96.5	The Original array is: [2.3, 6.1, 4.5, 96.5] Array after sorting is: [2.3, 4.5, 6.1, 96.5]	The Original array is: [2.3, 6.1, 4.5, 96.5] Array after sorting is: [2.3, 4.5, 6.1, 96.5]	✔

Passed all tests! ✔

Correct
Marks for this submission: 20.00/20.00.

Passed all tests! ✔

Correct
Marks for this submission: 20.00/20.00.

Write a python program to implement linear search on the given tuple of float values.

note: As the tuple is immutable convert the list to tuple to perform search

For example:

Input	Result
5 3.2 1.5 6.4 7.8 9.5 6.4	Tuple: 6.4 found
6 3.2 1.2 3.4 5.3 6.2 6.8 6.2	Tuple: 6.2 found

Answer: (penalty regime: 0 %)

```python
1  def search(t,n):
2      for i in range(len(t)):
3          if t[i]==n:
4              return i
5      return -1
6  t=[]
7  x=int(input())
8  for i in range(x):
9      t.append(input())
10 n=input()
11 res=search(t,n)
12 if res==-1:
13     print("Tuple:",n,"not found")
14 else:
15     print("Tuple:",n,"found")
```

	Input	Expected	Got	
✔	5 3.2 1.5 6.4 7.8 9.5 6.4	Tuple: 6.4 found	Tuple: 6.4 found	✔
✔	6 3.2 1.2 3.4 5.3 6.2 6.8 6.2	Tuple: 6.2 found	Tuple: 6.2 found	✔

	Input	Expected	Got	
✔	4 2.1 3.2 6.5 4.5 3.5	Tuple: 3.5 not found	Tuple: 3.5 not found	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Write a Python Program to print factorial of a number recursively.

For example:

Input	Result
5	Factorial of number 5 = 120
6	Factorial of number 6 = 720

Answer: (penalty regime: 0 %)

```
1  n=int(input())
2  fact=1
3  for i in range(1,n+1):
4      fact=fact*i
5  print("Factorial of number",n,"=",fact)
```

	Input	Expected	Got	
✔	5	Factorial of number 5 = 120	Factorial of number 5 = 120	✔
✔	6	Factorial of number 6 = 720	Factorial of number 6 = 720	✔
✔	7	Factorial of number 7 = 5040	Factorial of number 7 = 5040	✔
✔	8	Factorial of number 8 = 40320	Factorial of number 8 = 40320	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.