Started on	Friday, 11 April 2025, 2:02 PM
State	Finished
Completed on	Friday, 11 April 2025, 2:14 PM
Time taken	12 mins 47 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

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

For example:

Test	Input	Result
Merge_Sort(S)	6 4 2 3 1 6 5	The Original array is: [4, 2, 3, 1, 6, 5] Array after sorting is: [1, 2, 3, 4, 5, 6]
Merge_Sort(S)	5 2 6 4 3 1	The Original array is: [2, 6, 4, 3, 1] Array after sorting is: [1, 2, 3, 4, 6]

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
            S[k] = R[j]
            j += 1
            k += 1


n = int(input())
S= []
for i in range(n):
    l = int(input())
    S.append(l)

print("The Original array is: ",S)


Merge_Sort(S)

print("Array after sorting is: ",S)
```

	Test	Input	Expected	Got	
✔	Merge_Sort(S)	6 4 2 3 1 6 5	The Original array is: [4, 2, 3, 1, 6, 5] Array after sorting is: [1, 2, 3, 4, 5, 6]	The Original array is: [4, 2, 3, 1, 6, 5] Array after sorting is: [1, 2, 3, 4, 5, 6]	✔
✔	Merge_Sort(S)	5 2 6 4 3 1	The Original array is: [2, 6, 4, 3, 1] Array after sorting is: [1, 2, 3, 4, 6]	The Original array is: [2, 6, 4, 3, 1] Array after sorting is: [1, 2, 3, 4, 6]	✔

	Test	Input	Expected	Got	
✔	Merge_Sort(S)	4 3 5 6 1	The Original array is: [3, 5, 6, 1] Array after sorting is: [1, 3, 5, 6]	The Original array is: [3, 5, 6, 1] Array after sorting is: [1, 3, 5, 6]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 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 %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
def search(Tuple,x):
    for i in range(len(Tuple)):
        if Tuple[i]==x:
            return f"Tuple: {x} found"
    return f"Tuple: {x} not found"

Tuple=()
n=int(input())
for i in range(n):
    Tuple+=(float(input()),)
x=float(input())
print(search(Tuple,x))
```

	Input	Expected	Got	
✔	5 3.2 1.5 6.4 7.8 9.5 6.4	Tuple: 6.4 found	Tuple: 6.4 found	✔

	Input	Expected	Got	
✔	6 3.2 1.2 3.4 5.3 6.2 6.8 6.2	Tuple: 6.2 found	Tuple: 6.2 found	✔
✔	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.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a python program to implement quick sort on the given float values and print the sorted list and pivot value of each iteration.

For example:

Input	Result
5 2.3 3.2 1.6 4.2 3.9	Input List [2.3, 3.2, 1.6, 4.2, 3.9] pivot: 2.3 pivot: 3.2 pivot: 4.2 Sorted List [1.6, 2.3, 3.2, 3.9, 4.2]
4 5 2 49 3	Input List [5.0, 2.0, 49.0, 3.0] pivot: 5.0 pivot: 3.0 Sorted List [2.0, 3.0, 5.0, 49.0]

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
def quick_sort(alist,start,end):
    if end-start>1:
        p=partition(alist,start,end)
        quick_sort(alist,start,p)
        quick_sort(alist,p+1,end)

def partition(alist,start,end):
    pivot=alist[start]
    i=start+1
    j=end-1
    print("pivot: ",pivot)
    while True:
        while(i<=j and alist[i]<=pivot):
            i=i+1
        while(i<=j and alist[j]>=pivot):
            j=j-1

        if i<=j:
```

	Input	Expected	Got	
✔	5 2.3 3.2 1.6 4.2 3.9	Input List [2.3, 3.2, 1.6, 4.2, 3.9] pivot: 2.3 pivot: 3.2 pivot: 4.2 Sorted List [1.6, 2.3, 3.2, 3.9, 4.2]	Input List [2.3, 3.2, 1.6, 4.2, 3.9] pivot: 2.3 pivot: 3.2 pivot: 4.2 Sorted List [1.6, 2.3, 3.2, 3.9, 4.2]	✔
✔	4 5 2 49 3	Input List [5.0, 2.0, 49.0, 3.0] pivot: 5.0 pivot: 3.0 Sorted List [2.0, 3.0, 5.0, 49.0]	Input List [5.0, 2.0, 49.0, 3.0] pivot: 5.0 pivot: 3.0 Sorted List [2.0, 3.0, 5.0, 49.0]	✔

	Input	Expected	Got	
✔	6 3.1 4.2 5.1 2.3 7.4 5.9	Input List [3.1, 4.2, 5.1, 2.3, 7.4, 5.9] pivot: 3.1 pivot: 5.1 pivot: 7.4 Sorted List [2.3, 3.1, 4.2, 5.1, 5.9, 7.4]	Input List [3.1, 4.2, 5.1, 2.3, 7.4, 5.9] pivot: 3.1 pivot: 5.1 pivot: 7.4 Sorted List [2.3, 3.1, 4.2, 5.1, 5.9, 7.4]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a python program for a search function with parameter list name and the value to be searched on the given list of float values.

For example:

Test	Input	Result
search(List, n)	5 3.2 6.1 4.5 6.2 8.5 3.2	3.2 Found
search(List, n)	4 3.2 1.5 6.4 7.8 6.1	6.1 Not Found

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
def search(List, n):
    if n in List:
        print(f"{n} Found")
    else:
        print(f"{n} Not Found")

size = int(input())
List = [float(input()) for _ in range(size)]
n = float(input())
```

	Test	Input	Expected	Got	
✔	search(List, n)	5 3.2 6.1 4.5 6.2 8.5 3.2	3.2 Found	3.2 Found	✔
✔	search(List, n)	4 3.2 1.5 6.4 7.8 6.1	6.1 Not Found	6.1 Not Found	✔

	Test	Input	Expected	Got	
✔	search(List, n)	7 2.1 3.2 6.5 4.1 5.2 7.1 8.2 9.3	9.3 Not Found	9.3 Not Found	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Not answered

Mark 0.00 out of 20.00

Write a Python Program Using a recursive function to calculate the sum of a sequence
For example:

Input	Result
20	210
36	666
45	1035

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.