Started on	Wednesday, 19 March 2025, 9:29 AM
State	Finished
Completed on	Wednesday, 19 March 2025, 9:56 AM
Time taken	27 mins 2 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python Program to print the fibonacci series upto n_terms using Recursion.

For example:

Input	Result
10	Fibonacci series: 0 1 1 2 3 5 8 13 21 34
5	Fibonacci series: 0 1 1 2 3
7	Fibonacci series: 0 1 1 2 3 5 8

Answer: (penalty regime: 0 %)

```
def fibo(n):
    if n <=1:
        return n
    else:
        return fibo(n-2)+fibo(n-1)
def prfibo(nterm):
    for i in range(nterm):
        print(fibo(i))
nterm = int(input())
print("Fibonacci series:")
prfibo(nterm)
```

	Input	Expected	Got	
✔	10	Fibonacci series: 0 1 1 2 3 5 8 13 21 34	Fibonacci series: 0 1 1 2 3 5 8 13 21 34	✔
✔	5	Fibonacci series: 0 1 1 2 3	Fibonacci series: 0 1 1 2 3	✔
✔	7	Fibonacci series: 0 1 1 2 3 5 8	Fibonacci series: 0 1 1 2 3 5 8	✔
✔	9	Fibonacci series: 0 1 1 2 3 5 8 13 21	Fibonacci series: 0 1 1 2 3 5 8 13 21	✔
✔	11	Fibonacci series: 0 1 1 2 3 5 8 13 21 34 55	Fibonacci series: 0 1 1 2 3 5 8 13 21 34 55	✔

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 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 %)

```
def merge(S, temp, From, mid, to):
    a = From
    b = From
    c = mid + 1

    while b <= mid and c <= to:
        if S[b] < S[c]:
            temp[a] = S[b]
            b = b + 1
        else:
            temp[a] = S[c]
            c = c + 1
        a = a + 1

    while b <= mid:
        temp[a] = S[b]
        a = a + 1
        b = b + 1

    for b in range(From, to + 1):
        S[b] = temp[b]

```

	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]	✔

	Test	Input	Expected	Got	
✔	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.

Question **3**

Correct

Mark 20.00 out of 20.00

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

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

For example:

Input	Result
5 ram john akbar seetha oviya john	Tuple: john found
4 rohini fathima jenifer nizam rakesh	Tuple: rakesh not found

Answer: (penalty regime: 0 %)

```
n = int(input())
l= []
for i in range(n):
    ele = input()
    l.append(ele)
x = input()
tup = tuple(l)
temp = False
for k in tup:
    if(k == x):
        temp = True
        break
if(temp):
    print(f"Tuple: {x} found")
else:
    print(f"Tuple: {x} not found")
```

	Input	Expected	Got	
✔	5 ram john akbar seetha oviya john	Tuple: john found	Tuple: john found	✔

	Input	Expected	Got	
✔	4 rohini fathima jenifer nizam rakesh	Tuple: rakesh not found	Tuple: rakesh not found	✔
✔	6 rose jasmine tulips marigold hibiscus lotus lilly	Tuple: lilly not found	Tuple: lilly not found	✔

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 int values.

For example:

Test	Input	Result
search(List, n)	5 3 4 5 6 7 4	Found
search(List, n)	6 20 34 56 87 96 51 87	Found

Answer: (penalty regime: 0 %)

```
def search(List, n):
    for i in List:
        if(i==n):
            return 1
    else:
        return 0

if __name__=="__main__":
    g = int(input())
    List = [int(input()) for _ in range(g)]
    n = int(input())
    if(search(List,n)):
        print("Found")
    else:
        print("Not Found")
```

	Test	Input	Expected	Got	
✔	search(List, n)	5 3 4 5 6 7 4	Found	Found	✔

	Test	Input	Expected	Got	
✔	search(List, n)	6 20 34 56 87 96 51 87	Found	Found	✔
✔	search(List, n)	4 30 10 20 50 60	Not Found	Not Found	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

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 %)

```
def partition(arr, low, high):
    pivot = arr[low]
    print(f"pivot:  {pivot}")

    i = low + 1
    j = high

    while True:
        while i <= j and arr[i] <= pivot:
            i += 1

        while i <= j and arr[j] > pivot:
            j -= 1

        if i < j:
            swap(arr, i, j)
        else:
            break

    swap(arr, low, j)

    return 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]	✔

	Input	Expected	Got	
✔	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]	✔
✔	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.