Started on	Thursday, 20 March 2025, 11:17 AM
State	Finished
Completed on	Thursday, 20 March 2025, 11:26 AM
Time taken	9 mins 23 secs
Grade	**80.00** 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(m):
    if m <= 1:
        return m
    else:
        return fibo(m-2)+fibo(m-1)
def pfibo(term):
    for i in range(term):
        print(fibo(i))
term = int(input())
print("Fibonacci series:")
pfibo(term)
```

	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 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
    return 0

f = int(input())
List = []
for i in range(f):
    List.append(int(input()))
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 **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 %)

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

f = int(input())
k = []
for i in range(f):
    k.append(input())
n = input()
List = tuple(k)
if(search(List,n)):
    print(f"Tuple: {n} found")
else:
    print(f"Tuple: {n} not found")

```

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

	Input	Expected	Got	
✔	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 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]	✔
✔	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]	✔

	Test	Input	Expected	Got	
✔	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 **5**

Not answered

Mark 0.00 out of 20.00

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 %)

```
1
```