Started on	Friday, 14 March 2025, 2:46 PM
State	Finished
Completed on	Friday, 14 March 2025, 3:41 PM
Time taken	54 mins 46 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 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 %)

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)

num = int(input())
print(f"Factorial of number {num} = {factorial(num)}")

```

	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.

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_Sort(S):
    n = len(S)
    current_size = 1

    while current_size < n:
        left = 0
        while left < n - 1:
            mid = min(left + current_size - 1, n - 1)
            right = min(left + 2 * current_size - 1, n - 1)

            merge(S, left, mid, right)
            left += 2 * current_size

        current_size *= 2


def merge(S, left, mid, right):
    n1 = mid - left + 1
    n2 = right - mid

    L = [S[left + i] for i in range(n1)]
    R = [S[mid + 1 + i] for i in range(n2)]
```

	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 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 %)

```
def length(k, n):
    for i in k:
        if i == n:
            print("Tuple:", n, "found")
            return
    print("Tuple:",n, "not found")

arr = int(input())
k = []
for _ in range(arr):
    k.append(input())
n = input()
length(k, n)
```

	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 **4**

Correct

Mark 20.00 out of 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 %)

```
def binarySearchAppr(arr, low,high, x):
    while(low<=high):
        mid=(low+high)//2
        if arr[mid]==x:
            return mid
        elif x<arr[mid]:
            high=mid-1
        else:
            low=mid+1
    return -1
n=int(input());
arr=[];
for i in range(n):
    arr.append(input())
x=input()
arr.sort();
result=binarySearchAppr(arr,0,len(arr)-1,x)
if result>=0:
    print("Element is present at index",result);
else:
    print("Element is not present in array");
```

	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	✔

	Test	Input	Expected	Got	

✔	`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`	✔

✔	`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.

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]
    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:
            arr[i], arr[j] = arr[j], arr[i]
        else:
            break

    arr[low], arr[j] = arr[j], arr[low]
    return j

def quick_sort(arr, low, high):
```

	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.