Question **1**

Correct

Mark 20.00 out of 20.00

Flag question

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,l,r,x):
    while(l <= r):
        mid = (l +r ) // 2
        if x == arr[mid]:
            return mid
        elif x < arr[mid]:
            r = mid - 1
        else:
            l = mid + 1
    return -1

arr = sorted([input() for i in range(int(input()))])
x = input()

rr = binarySearchAppr(arr,0,len(arr) - 1,x)
if rr >= 0:
    print(f"Element is present at index {rr}")
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`
	`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.

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

For example:

Question **2**

Correct

Mark 20.00 out of 20.00

Flag question

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 %)

```
def qsort(L):
    if L==[]:
        return[]
    pivot=L[0:1]
    left=qsort([x for x in L[1:]if x<L[0]])
    right=qsort([x for x in L[1:]if x>=L[0]])
    print("left: ",left)
    print("right: ",right)
    return left+pivot+right
list1=[]
n=int(input())
for i in range(n):
    list1.append(float(input()))
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: []	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: []	

	Input	Expected	Got	
		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]	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.

Question **3**

Correct

Mark 20.00 out of 20.00

Flag question

Write a python program to implement merge sort without using recursive function on the given list of values.

For example:

Input	Result
7 33 42 9 37 8 47 5	left: [33] Right: [42] left: [9] Right: [37] left: [8] Right: [47] left: [5] Right: [] left: [33, 42] Right: [9, 37] left: [8, 47] Right: [5] left: [9, 33, 37, 42] Right: [5, 8, 47] [5, 8, 9, 33, 37, 42, 47]
6 10 3 5 61 74 92	left: [10] Right: [3] left: [5] Right: [61] left: [74] Right: [92] left: [3, 10] Right: [5, 61] left: [74, 92] Right: [] left: [3, 5, 10, 61] Right: [74, 92] [3, 5, 10, 61, 74, 92]

Answer: (penalty regime: 0 %)

```
def merge(left, right):
    result = []
    x, y = 0, 0
    for k in range(0, len(left) + len(right)):
        if x == len(left):
            result.append(right[y])
            y += 1
        elif y == len(right):
            result.append(left[x])
            x += 1
        elif right[y] < left[x]:
            result.append(right[y])
            y += 1
        else:
            result.append(left[x])
            x += 1
    return result
def mergesort(ar_list):
    length = len(ar_list)
    size = 1
    while size < length:
        size+=size
```

	Input	Expected	Got	
	7 33 42 9 37 8 47 5	left: [33] Right: [42] left: [9] Right: [37] left: [8] Right: [47] left: [5] Right: [] left: [33, 42] Right: [9, 37]	left: [33] Right: [42] left: [9] Right: [37] left: [8] Right: [47] left: [5] Right: [] left: [33, 42] Right: [9, 37]	

	Input	Expected	Got	
		left: [8, 47] Right: [5] left: [9, 33, 37, 42] Right: [5, 8, 47] [5, 8, 9, 33, 37, 42, 47]	left: [8, 47] Right: [5] left: [9, 33, 37, 42] Right: [5, 8, 47] [5, 8, 9, 33, 37, 42, 47]	
	6 10 3 5 61 74 92	left: [10] Right: [3] left: [5] Right: [61] left: [74] Right: [92] left: [3, 10] Right: [5, 61] left: [74, 92] Right: [] left: [3, 5, 10, 61] Right: [74, 92] [3, 5, 10, 61, 74, 92]	left: [10] Right: [3] left: [5] Right: [61] left: [74] Right: [92] left: [3, 10] Right: [5, 61] left: [74, 92] Right: [] left: [3, 5, 10, 61] Right: [74, 92] [3, 5, 10, 61, 74, 92]	
	5 4 12 6 98 3	left: [4] Right: [12] left: [6] Right: [98] left: [3] Right: [] left: [4, 12] Right: [6, 98] left: [3] Right: [] left: [4, 6, 12, 98] Right: [3] [3, 4, 6, 12, 98]	left: [4] Right: [12] left: [6] Right: [98] left: [3] Right: [] left: [4, 12] Right: [6, 98] left: [3] Right: [] left: [4, 6, 12, 98] Right: [3] [3, 4, 6, 12, 98]	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Flag question

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-1)+fibo(n-2)

n = int(input())
print("Fibonacci series:")
for i in range(n):
    print(fibo(i))
```

	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 **5**

Correct

Mark 20.00 out of 20.00

Flag question

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:
            print("Found")
            return
    print("Not Found")

List = [(input()) for i in range(int(input()))]
n = (input())
```

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