
Started on Saturday, 26 April 2025, 9:58 AM

State Finished

Completed on Saturday, 26 April 2025, 2:47 PM

Time taken 4 hours 49 mins

Overdue 2 hours 49 mins

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 %)

```
1 def fibonacci(n):
2     if n <= 0:
3         return 0
4     elif n == 1:
5         return 1;
6     else:
7         return fibonacci(n-1) + fibonacci(n-2)
8
9 def print_fibonacci_series(n_terms):
10     print("Fibonacci series:")
11     for i in range(n_terms):
12         print(fibonacci(i))
13
14 n_terms = int(input())
15 print_fibonacci_series(n_terms)
```

	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 binary search on the given list of float values using iterative method

For example:

Test	Input	Result
binarySearchAppr(arr, 0, len(arr)-1, x)	5 3.2 6.1 4.5 9.6 8.3 6.1	Element is present at index 2
binarySearchAppr(arr, 0, len(arr)-1, x)	6 3.1 2.3 5.1 4.6 3.2 9.5 4.6	Element is present at index 3

Answer: (penalty regime: 0 %)

```

1 def binarySearchAppr (arr, start, end, x):
2     if end >= start:
3         mid = (start +end)//2
4         if arr[mid] == x:
5             return mid
6         elif arr[mid] > x:
7             return binarySearchAppr(arr, start, mid-1, x)
8         else:
9             return binarySearchAppr(arr,mid+1,end,x)
10    else:
11        return -1
12 arr=[]
13 n=int(input())
14 for i in range(n):
15     arr.append(input())
16 arr = sorted(arr)
17 x =input()
18 result = binarySearchAppr(arr,0,len(arr)-1,x)
19 if result != -1:
20     print ("Element is present at index "+str(result))
21 else:
22     print ("Element is not present in array")

```

	Test	Input	Expected	Got	
✓	binarySearchAppr(arr, 0, len(arr)-1, x)	5 3.2 6.1 4.5 9.6 8.3 6.1	Element is present at index 2	Element is present at index 2	✓
✓	binarySearchAppr(arr, 0, len(arr)-1, x)	6 3.1 2.3 5.1 4.6 3.2 9.5 4.6	Element is present at index 3	Element is present at index 3	✓

	Test	Input	Expected	Got	
✓	binarySearchAppr(arr, 0, len(arr)-1, x)	8 2.1 6.3 5.2 4.2 9.3 6.7 5.6 9.8 7.2	Element is not present in array	Element is not present in array	✓

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 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 %)

```

1 def search(List,n):
2     for i in List:
3         if i==n:
4             print(n,"Found")
5             break
6     else:
7         print(n,"Not Found")
8 s=int(input())
9 List=[input() for i in range(s)]
10 n=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 4

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

Answer: (penalty regime: 0 %)

```

1 def quicksort(arr, low, high):
2     if low < high:
3         pi = partition(arr, low, high)
4         quicksort(arr, low, pi - 1)
5         quicksort(arr, pi + 1, high)
6
7 def partition(arr, low, high):
8     pivot = arr[low]
9     left = low + 1
10    right = high
11
12    while True:
13        while left <= right and arr[left] <= pivot:
14            left = left + 1
15        while left <= right and arr[right] >= pivot:
16            right = right - 1
17        if left <= right:
18            arr[left], arr[right] = arr[right], arr[left]
19        else:
20            break
21
22    arr[low], arr[right] = arr[right], arr[low]
```

	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 **5**

Not answered

Mark 0.00 out of 20.00

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

For example:

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

1 ||