

- a) Write a program in Python that makes use of the above implementation as a function. Your program should input the array from the user. Also compare the execution time of function with built-in search function available and write down the results.

```
1 import timeit # for measuring execution time
2 data = eval(input("Enter data in list: ")) # Taking input in list format
3 item = int(input("Enter item to be searched: "))
4 data.sort() # sorting data
5 def Binary_search(data, item):
6     beg = 0
7     end = len(data) - 1
8
9     while beg <= end:
10         mid = (beg + end) // 2
11         if data[mid] == item:
12             return mid # returning position of item
13         elif item < data[mid]:
14             end = mid - 1
15         else:
16             beg = mid + 1
17     return None
18 print("Measuring Execution time ")
19 print("Using own function")
20 result = Binary_search(data, item)
21 if result is not None:
22     print("Location of item is:", result)
23 else:
24     print("location of item is Null")
25 print(timeit.timeit(lambda: Binary_search(data, item), number=10000), "Seconds")
26 print("\nUsing built-in function")
27 print("Location of item is: ", data.index(item))
28 print(timeit.timeit(lambda: data.index(item), number=10000), "Seconds")
29
30 Output:
31
32 Enter data in list: [1,2,3,4,5,6,7,8,9,10]
33 Enter item to be searched: 4
34 Measuring Execution time
35 Using own function
36 Location of item is: 3
37 0.00809320000007574 Seconds
38
39 Using built-in function
40 Location of item is: 3
41 0.0022168999994391925 Seconds
```

Output:

```
Enter data in list: [1,2,3,4,5,6,7,8,9,10]
Enter item to be searched: 4
Measuring Execution time
Using own function
Location of item is: 3
0.00809320000007574 Seconds

Using built-in function
Location of item is: 3
0.0022168999994391925 Seconds
```


c) Construct a function to take input of an array from the user and prompt the user if he enters unsorted data. Modify the *binary search* algorithm presented here so as to insert the element in the array if the search remains unsuccessful. Make sure that the array remains sorted after the insertion. Implement the revised algorithm in Python.

```
1 import timeit # for measuring execution time
2 data = eval(input("Enter data in list: ")) # Taking input in list format
3 while data != sorted(data):
4     print("Data is not sorted, Please enter sorted data")
5     data = eval(input("Enter data in list: ")) # Prompt user again
6 print("Data is sorted")
7 item = int(input("Enter item to be searched: "))
8 if item not in data:
9     print("Item not in data, Inserting item in data...")
10    data.append(item)
11    data.sort()
12    print(f"{item} inserted in data and data is sorted")
13
14 def Binary_search(data, item):
15     beg = 0
16     end = len(data) - 1
17
18     while beg <= end:
19         mid = (beg + end) // 2
20         if data[mid] == item:
21             return mid # returning position of item
22         elif item < data[mid]:
23             end = mid - 1
24         else:
25             beg = mid + 1
26     return None
27 print("Measuring Execution time ")
28 print("Using own function")
29 result = Binary_search(data, item)
30 if result is not None:
31     print("Location of item is:", result)
32 else:
33     print("location of item is Null")
34 print(timeit.timeit(lambda: Binary_search(data, item), number=10000), "Seconds")
35 print("\nUsing built-in function")
36 print("Location of item is: ", data.index(item))
37 print(timeit.timeit(lambda: data.index(item), number=10000), "Seconds")
```

Output:

```
Enter data in list: [1,2,3,4]
Data is sorted
Enter item to be searched: 5
Item not in data, Inserting item in data...
5 inserted in data and data is sorted
Measuring Execution time
Using own function
Location of item is: 4
0.008548599998903228 Seconds

Using built-in function
Location of item is: 4
0.0030119000002741814 Seconds
```