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.

```
import timeit # for measuring execution time
data = eval(input("Enter data in list: ")) # Taking input in list format
item = int(input("Enter item to be searched: "))
data.sort() # sorting data
def Binary_search(data, item):
    beg = 0
    end = len(data) - 1

    while beg <= end:
        mid = (beg + end) // 2
        if data[mid] == item:
            return mid # returning position of item
        elif item < data[mid]:
            end = mid - 1
        else:
            beg = mid + 1
    return None
print("Measuring Execution time ")
print("Using own function")
result = Binary_search(data, item)
if result is not None:
    print("Location of item is:", result)
else:
    print("location of item is Null")
print(timeit.timeit(lambda: Binary_search(data, item), number=10000), "Seconds")
print("\nUsing built-in function")
print("Location of item is: ", data.index(item))
print(timeit.timeit(lambda: data.index(item), number=10000), "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
0.0080932000007574 Seconds

Using built-in function
Location of item is:  3
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
0.0080932000007574 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.

```python
import timeit # for measuring execution time
data = eval(input("Enter data in list: ")) # Taking input in list format
while data != sorted(data):
    print("Data is not sorted, Please enter sorted data")
    data = eval(input("Enter data in list: ")) # Prompt user again
print("Data is sorted")
item = int(input("Enter item to be searched: "))
if item not in data:
    print("Item not in data, Inserting item in data...")
    data.append(item)
    data.sort()
    print(f"{item} inserted in data and data is sorted")

def Binary_search(data, item):
    beg = 0
    end = len(data) - 1

    while beg <= end:
        mid = (beg + end) // 2
        if data[mid] == item:
            return mid # returning position of item
        elif item < data[mid]:
            end = mid - 1
        else:
            beg = mid + 1
    return None
print("Measuring Execution time ")
print("Using own function")
result = Binary_search(data, item)
if result is not None:
    print("Location of item is:", result)
else:
    print("location of item is Null")
print(timeit.timeit(lambda: Binary_search(data, item), number=10000), "Seconds")
print("\nUsing built-in function")
print("Location of item is: ", data.index(item))
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
0.0085485999989032 28 Seconds

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