You can view this report online at : https://www.hackerrank.com/x/tests/1528818/candidates/50260563/report

Full Name:	Breeha Qasim
Email:	bq08283@st.habib.edu.pk
Test Name:	**CS 102 - Lab 6 - Spring 2023**
Taken On:	17 Feb 2023 10:30:19 PKT
Time Taken:	113 min 12 sec/ 2880 min
Work Experience:	< 1 years
Invited by:	Aisha
Skills Score:	
Tags Score:	

98.9%

692/700

scored in **CS 102 - Lab 6 - Spring 2023** in 113 min 12 sec on 17 Feb 2023 10:30:19 PKT

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review.

	Question Description	Time Taken	Score	Status
Q1	Binary Search Iterative > Coding	7 min 10 sec	100/ 100	
Q2	Binary Search Iterative Modified > Coding	23 min 13 sec	92/ 100	
Q3	Binary Search Recursive > Coding	9 min 11 sec	100/ 100	
Q4	Binary Search Recursive Modified > Coding	1 min 36 sec	100/ 100	
Q5	Updating Student Record > Coding	46 min 1 sec	100/ 100	
Q6	Length of the Line > Coding	14 min 24 sec	100/ 100	
Q7	Finding Multiple > Coding	11 min 19 sec	100/ 100	

QUESTION 1

Needs Review

Score 100

Binary Search Iterative > Coding

QUESTION DESCRIPTION

Write a function named `binary_search_iterative`, which takes the following two inputs:

1. `list` – An input list (sorted data)

2. `item` – An item to search in the list

and return the index of `item`; -1 otherwise.

```
>>> binary_search_iterative( [0, 1, 2, 8, 13, 17, 19, 32, 42], 3)
-1

>>> binary_search_iterative(  [0, 1, 2, 8, 13, 17, 19, 32, 42], 17)
5
```

INTERVIEWER GUIDELINES

```
def binary_search_iterative(lst,item):
    low = 0
    high = len(lst)-1
    i = (low + high)//2
    j = low + high
    while j>0:
        if lst[i]==item:
            return i
        elif lst[i]<item:
            low = i+1
        elif lst[i]>item:
            high = i-1
        i = (low + high)//2
        j = j//2
    return -1
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1  def binary_search_iterative(lst,item):
2      high=len(lst)-1
3      low=0
4      mid=0
5      while low<=high:
6          mid=(high+low)//2
7          if lst[mid]<item:
8              low=mid+1
9          elif lst[mid]>item:
10             high=mid-1
11         else:
12             return mid
13     return -1
14
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	6	0.107 sec	9.08 KB
Testcase 1	Easy	Sample case	Success	6	0.0618 sec	8.91 KB
Testcase 2	Easy	Hidden case	Success	8	0.0704 sec	9.11 KB
Testcase 3	Easy	Hidden case	Success	8	0.0466 sec	9.07 KB
Testcase 4	Easy	Hidden case	Success	8	0.0761 sec	9.31 KB
Testcase 5	Easy	Hidden case	Success	8	0.0546 sec	9.24 KB

Testcase 6	Easy	Hidden case	Success	8	0.0623 sec	9.08 KB
Testcase 7	Easy	Hidden case	Success	8	0.0572 sec	9.23 KB
Testcase 8	Easy	Hidden case	Success	8	0.0551 sec	8.91 KB
Testcase 9	Easy	Hidden case	Success	8	0.0871 sec	9.18 KB
Testcase 10	Easy	Hidden case	Success	8	0.0527 sec	8.95 KB
Testcase 11	Easy	Hidden case	Success	8	0.0612 sec	9.05 KB
Testcase 12	Easy	Hidden case	Success	8	0.0652 sec	9.29 KB

No Comments

QUESTION 2

Correct Answer

Score 92

Binary Search Iterative Modified › Coding

QUESTION DESCRIPTION

Create a function named `binary_search_iterative_modified` which takes the following two inputs:

1. `list` – An input list (sorted data)
2. `item` – An item to search in the list

and search for the `item` in the list. If item is not found, add the item to the list and return its index.

```
>>> binary_search_iterative_modified([0, 1, 2, 8, 13, 17, 19, 32, 42], 8)
3

>>> binary_search_iterative_modified([0, 1, 2, 3, 8, 13, 17, 19, 32, 42],
-1)
0
```

INTERVIEWER GUIDELINES

```
def binary_search_iterative_modified(lst,item):
    low = 0
    high = len(lst)-1
    i = (low + high)//2
    j = low + high
    while j>0:
        if lst[i]==item:
            return i
        elif lst[i]<item:
            low = i+1
        elif lst[i]>item:
            high = i-1
        i = (low + high)//2
        j = j//2
    lst.insert(low,item)
    return low
```

CANDIDATE ANSWER

Language used: **Python 3**

```
'''def binary_search_iterative(lst,item):
    high=len(lst)-1
    low=0
    mid=0
```

```
    while low<=high:
        mid=(high+low)//2
        if lst[mid]<item:
            low=mid+1
        elif lst[mid]>item:
            high=mid-1
        else:
            return mid
    return -1'''
def binary_search_iterative_modified(lst,item):
    high=len(lst)-1
    low=0
    mid=0
    while low<=high:
        mid=(high+low)//2
        if lst[mid]<item:
            low=mid+1
        elif lst[mid]>item:
            high=mid-1
        else:
            return mid
    if item<mid:
        lst.insert(mid,item)
        return mid
    elif item>mid:
        lst.insert(mid+1,item)
        return mid+1
    '''found=binary_search_iterative(lst,item)
    if found==-1:
        lst.append(item)
        lst.sort()
        return binary_search_iterative(lst,item)
    else:
        return found'''
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	6	0.0753 sec	9.02 KB
Testcase 1	Easy	Sample case	Success	6	0.0406 sec	9.11 KB
Testcase 2	Easy	Hidden case	Success	8	0.0776 sec	8.8 KB
Testcase 3	Easy	Hidden case	Success	8	0.0839 sec	9.09 KB
Testcase 4	Easy	Hidden case	Success	8	0.0532 sec	8.93 KB
Testcase 5	Easy	Hidden case	Success	8	0.0601 sec	8.91 KB
Testcase 6	Easy	Hidden case	Success	8	0.0515 sec	8.97 KB
Testcase 7	Easy	Hidden case	Success	8	0.0601 sec	9.04 KB
Testcase 8	Easy	Hidden case	Success	8	0.0443 sec	9.27 KB
Testcase 9	Easy	Hidden case	Success	8	0.0384 sec	9.09 KB
Testcase 10	Easy	Hidden case	Success	8	0.0485 sec	9.28 KB
Testcase 11	Easy	Hidden case	Wrong Answer	0	0.0738 sec	8.99 KB
Testcase 12	Easy	Sample case	Success	8	0.0459 sec	9.03 KB

No Comments

QUESTION 3

Correct Answer

Score 100

Binary Search Recursive › Coding

QUESTION DESCRIPTION

Implement the binary search algorithm with recursive approach.
Write a function named `binary_search_recursive`, which takes the following inputs:

1. `list` – An input list (sorted data)
2. `item` – An item to search in the list
3. `low` – Low index of data list
4. `high` – High index of data list

and return the index of `item`; -1 otherwise.

```
>>> binary_search_recursive( [0, 1, 2, 8, 13, 17, 19, 32, 42], 19, 0, 8)
6

>>> binary_search_recursive( [0, 1, 2, 8, 13, 17, 19, 32, 42], 3, 0, 8)
-1
```

INTERVIEWER GUIDELINES

```
def binary_search_recursive(lst, item, low, high):
    if low > high :
        return -1
    elif lst[(low+high)//2]==item:
        return (low+high)//2
    elif lst[(low+high)//2]<item:
        return binary_search_recursive(lst, item, ((low+high)//2)+1,
high)
    else:
        return binary_search_recursive(lst, item, low, ((low+high)//2)-1)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1  def binary_search_recursive(lst,item,low,high):
2      mid=(high+low)//2
3      if lst[mid] == item:
4          return mid
5      elif low > high:
6          return -1
7      else:
8          if lst[mid]<item:
9              return binary_search_recursive(lst,item,mid+1,high)
10         #low=mid+1
11         else:
12             return binary_search_recursive(lst,item,low,mid-1)
13
14
15
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	9	0.0524 sec	9.07 KB
Testcase 1	Easy	Sample case	Success	9	0.065 sec	8.98 KB

Testcase 2	Easy	Hidden case	Success	9	0.0702 sec	9.11 KB
Testcase 3	Easy	Hidden case	Success	9	0.1339 sec	8.91 KB
Testcase 4	Easy	Hidden case	Success	9	0.0584 sec	9.31 KB
Testcase 5	Easy	Hidden case	Success	9	0.0696 sec	8.88 KB
Testcase 6	Easy	Hidden case	Success	9	0.0891 sec	8.91 KB
Testcase 7	Easy	Hidden case	Success	9	0.0676 sec	9.03 KB
Testcase 8	Easy	Hidden case	Success	9	0.0475 sec	9.19 KB
Testcase 9	Easy	Hidden case	Success	10	0.0467 sec	8.91 KB
Testcase 10	Easy	Sample case	Success	9	0.0649 sec	9.27 KB

No Comments

QUESTION 4

Correct Answer

Score 100

Binary Search Recursive Modified › Coding

QUESTION DESCRIPTION

Write a function named `binary_search_recursive_modified`, which takes the following inputs:

1. `list` – An input list (sorted data)
2. `item` – An item to search in the list
3. `low` – Low index of data list
4. `high` – High index of data list

and search for the `item` in the `list`. If item is not found, add the item to the list and return its index.

```
>>> binary_search_recursive_modified([0, 1, 2, 8, 13, 17, 19, 32, 42], 3,
0, 8)
3

>>> binary_search_recursive_modified([0, 1, 2, 8, 13, 17, 19, 32, 42], 17,
0, 8)
5
```

INTERVIEWER GUIDELINES

```
def binary_search_recursive_modified(lst, item, low, high):
    if low > high :
        lst.insert(low,item)
        return low
    elif lst[(low+high)//2]==item:
        return (low+high)//2
    elif lst[(low+high)//2]<item:
        return binary_search_recursive_modified(lst, item,
((low+high)//2)+1, high)
    elif lst[(low+high)//2]>item:
        return binary_search_recursive_modified(lst, item, low,
((low+high)//2)-1)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 def binary_search_recursive_modified(lst,item,low,high):
2     mid=(high+low)//2
```

```
    if lst[mid] == item:
        return mid
    elif low > high:
        return mid+1
    else:
        if lst[mid]<item:
            return binary_search_recursive_modified(lst,item,mid+1,high)
        #low=mid+1
        else:
            return binary_search_recursive_modified(lst,item,low,mid-1)


```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	8	0.0443 sec	9.15 KB
Testcase 1	Easy	Sample case	Success	8	0.0389 sec	9.14 KB
Testcase 2	Easy	Hidden case	Success	8	0.0594 sec	9.07 KB
Testcase 3	Easy	Hidden case	Success	8	0.1002 sec	9.09 KB
Testcase 4	Easy	Hidden case	Success	8	0.0399 sec	8.98 KB
Testcase 5	Easy	Hidden case	Success	8	0.0517 sec	9.02 KB
Testcase 6	Easy	Hidden case	Success	8	0.038 sec	9.12 KB
Testcase 7	Easy	Hidden case	Success	8	0.0857 sec	8.87 KB
Testcase 8	Easy	Hidden case	Success	8	0.0349 sec	8.96 KB
Testcase 9	Easy	Hidden case	Success	10	0.0915 sec	9.16 KB
Testcase 10	Easy	Hidden case	Success	10	0.0605 sec	8.96 KB
Testcase 11	Easy	Sample case	Success	8	0.0401 sec	9.1 KB

No Comments

QUESTION 5

Correct Answer

Score 100

Updating Student Record › Coding

QUESTION DESCRIPTION

Using binary search approach, write a function named `update_record`, which takes the following inputs:

1. `students_records` – Nested List of Tuples. Each tuple of the list represents student's data.
2. `ID` – An ID of a student whose data has to be updated
3. `record_title` – type of data that has to be updated
4. `data` – A new data which should replace record_title data

and update the record of students' data associated to specific ID.

If **ID** is given as the data to be updated, then return a message that ID cannot be updated.
If **ID** is not found in students_records, then return a message that Record not found.

NOTE: The type of record_title input can be "`ID`", "`Email`", "`Mid1`" or "`Mid2`". Please use the same spelling for these types in your code, because the same have been used in test cases.

```
student_records = [("aa02822", "ea02822", 80, 65),
 ("ea02822", "ea02822@st.habib.edu.pk", 80, 65),
```

```
 ("fa08877", "fa08877@st.habib.edu.pk", 66, 67),
 ("gh04588", "gh04588@st.habib.edu.pk", 33, 50)]
```

```
>>> update_record(students_records, "aa02822", "ID", "aa02456")
ID cannot be updated

>>> update_record(students_records, "aa02822", "Email",
"aa02822@st.habib.edu.pk")
[("aa02822", "aa02822@st.habib.edu.pk", 80, 65),
 ("ea02822", "ea02822@st.habib.edu.pk", 80, 65),
 ("fa08877", "fa08877@st.habib.edu.pk", 66, 67),
 ("gh04588", "gh04588@st.habib.edu.pk", 33, 50)]

>>> update_record(students_records, 'f08877', "Mid2", 50)
Record not found
```

CANDIDATE ANSWER

Language used: **Python 3**

```
def binary_search_iterative(lst,item):
    high=len(lst)-1
    low=0
    mid=0
    while low<=high:
        mid=(high+low)//2
        if lst[mid]<item:
            low=mid+1
        elif lst[mid]>item:
            high=mid-1
        else:
            return mid
    return -1
def update_record(student_records,ID,record_title,data):
    if record_title=="ID":
        return "ID cannot be updated"
    ID_lst=[]
    for i in student_records:
        ID_lst.append(i[0])
    found=binary_search_iterative(ID_lst,ID)
    if found==-1:
        return "Record not found"

    if record_title=="Email":
        tup=(student_records[found][0],data,student_records[found]
[2],student_records[found][3])
        student_records[found]=tup
        return student_records

    if record_title=="Mid1":
        tup=(student_records[found][0],student_records[found]
[1],int(data),student_records[found][3])
        student_records[found]=tup
        return student_records

    if record_title=="Mid2":
        tup=(student_records[found][0],student_records[found]
[1],student_records[found][2],int(data))
```

```
        student_records[found]=tup
        return student_records
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	20	0.073 sec	8.86 KB
Testcase 1	Easy	Sample case	Success	20	0.0426 sec	9.29 KB
Testcase 2	Easy	Sample case	Success	20	0.0365 sec	9.1 KB
Testcase 3	Easy	Sample case	Success	20	0.0782 sec	9.04 KB
Testcase 4	Easy	Sample case	Success	20	0.0875 sec	9.18 KB

No Comments

QUESTION 6

Needs Review

Score 100

Length of the Line › Coding

QUESTION DESCRIPTION

Using Binary Search, write a function named `length_of_line`, which takes two inputs:

1. `points_lists` – The list containing nested lists of starting and ending x and y coordinates of different lines. For e.g: [[(x1, y1), (x2, y2)], [(x1, y1), (x2, y2)], [(x1, y1), (x2, y2)], [(x1, y1), (x2, y2)]].
2. `length` – The length of a line.

Now determine, if any line present in the points_lists has a same length as that of required length.
Return the index of that line. If the line is not found, return -1.

Hint: Estimate the length of a line with the help of distance formula.

```
>>> length_of_line([[(2,4),(4,6)], [(1,2),(4,6)], [(4,0),(6, 6)]], 5.0)
1

>>> length_of_line([ [(2,4),(4,6)], [(1,2),(4,6)],[(4,0),(6, 6)]], 6.32)
2

>>> length_of_line([ [(2,4),(4,6)], [(1,2),(4,6)],[(4,0),(6, 6)]], 1.0)
-1
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1  import math
2  def binary_search_iterative(lst,item):
3      high=len(lst)-1
4      low=0
5      mid=0
6      while low<=high:
7          mid=(high+low)//2
8          if lst[mid]<item:
9              low=mid+1
10         elif lst[mid]>item:
11             high=mid-1
12         else:
```

```
            return mid
    return -1
def length_of_line(points_list,length):
    dist_srch=[]
    for i in points_list:
        dist=math.sqrt((i[1][0]-i[0][0])**2+(i[1][1]-i[0][1])**2)
        dist_srch.append(round(dist,2))
    return binary_search_iterative(dist_srch,length)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	25	0.0531 sec	9.06 KB
Testcase 1	Easy	Sample case	Success	25	0.0533 sec	8.99 KB
Testcase 2	Easy	Sample case	Success	25	0.0621 sec	9.11 KB
Testcase 3	Easy	Sample case	Success	25	0.0626 sec	8.95 KB

No Comments

QUESTION 7

Needs Review

Score 100

Finding Multiple › Coding

QUESTION DESCRIPTION

Using binary and linear search approaches, write a function named `finding_multiple`, which takes the following inputs:

1. `lst` – list of items (sorted data)
2. `item` – the item needs to be searched in the `lst`

and return the list of all the indexes where the item is found in the `lst`, otherwise return empty list.

```
>>> finding_multiple([0, 1, 2, 8, 13, 17, 17, 17 ,17, 19, 32, 42], 17)
[5, 6, 7, 8]
>>> finding_multiple([0, 1, 2, 8, 13, 17, 17, 17 ,17, 19, 32, 42], 34)
[]
```

CANDIDATE ANSWER

Language used: **Python 3**

```

def finding_multiple(lst,item):
    high=len(lst)-1
    low=0
    mid=0
    final=[]
    while low<=high:
        mid=(high+low)//2
        if lst[mid]<item:
            low=mid+1
        elif lst[mid]>item:
            high=mid-1
        else:
            m=mid
            while m<len(lst) and lst[m]==item:
                final.append(m)
```

```
17                m+=1
18            m=mid-1
19            while m>=0 and lst[m]==item:
20                final.insert(0,m)
21                m-=1
22            return final
23    return []
24
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	15	0.0478 sec	9.27 KB
Testcase 1	Easy	Sample case	Success	17	0.0658 sec	8.88 KB
Testcase 2	Easy	Hidden case	Success	17	0.0532 sec	8.92 KB
Testcase 3	Easy	Hidden case	Success	17	0.0718 sec	8.95 KB
Testcase 4	Easy	Sample case	Success	17	0.0563 sec	8.91 KB
Testcase 5	Easy	Hidden case	Success	17	0.0715 sec	8.9 KB

No Comments

PDF generated at: 19 Apr 2023 10:40:52 UTC