You can view this report online at : https://www.hackerrank.com/x/tests/1533113/candidates/50460464/report

Full Name:	Breeha Qasim
Email:	bq08283@st.habib.edu.pk
Test Name:	**CS102 - Lab 7 - Spring 2023**
Taken On:	24 Feb 2023 11:23:51 PKT
Time Taken:	2079 min 28 sec/ 3000 min
Work Experience:	< 1 years
Invited by:	Aisha
Skills Score:	
Tags Score:	

100%

570/570

scored in **CS102 - Lab 7 - Spring 2023** in 2079 min 28 sec on 24 Feb 2023 11:23:51 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	Selection Sort > Coding	24 min 29 sec	140/ 140	!
Q2	Insertion Sort > Coding	19 min 20 sec	130/ 130	✓
Q3	Bubble Sort > Coding	19 min 59 sec	140/ 140	!
Q4	Sort Matrix By Row > Coding	16 min 32 sec	60/ 60	!
Q5	Sort Matrix By Column Number > Coding	11 min 3 sec	60/ 60	!
Q6	Sorting Rectangle Records > Coding	6 min 17 sec	40/ 40	✓

QUESTION 1

Needs Review

Score 140

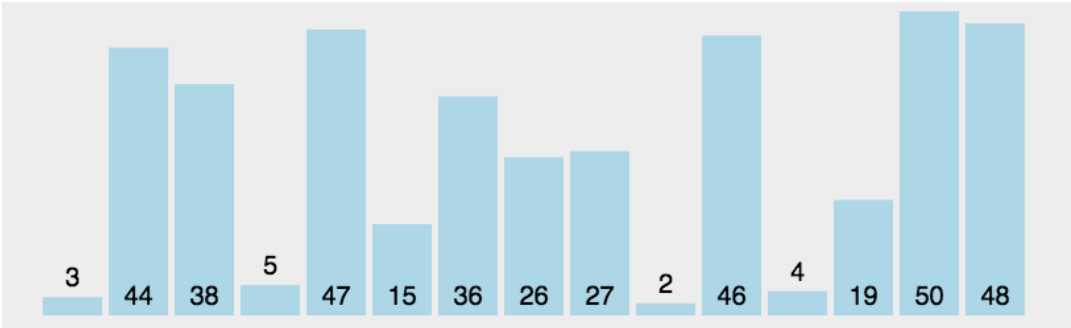
Selection Sort > Coding

QUESTION DESCRIPTION

It is one of the simplest sorting algorithm. We repeatedly find the next largest (or smallest) element in the array and move it to its final position in the sorted array. Assume that we wish to sort the array in increasing order, i.e. the smallest element at the beginning of the array and the largest element at the end. We begin by selecting the smallest element and moving it to the first position. We can do this by swapping the

smallest element at the first index of array. We then reduce the effective size of the array by one element and repeat the process on the smaller (sub)array. The process stops when the effective size of the array becomes 1 (an array of 1 element is already sorted).

Sorting Algorithm Simulation: https://visualgo.net/bn/sortin



Write a function `selection_sort(list)` that take a `list` and sort it using **Selection Sort Algorithm**. Print each pass or iteration.

```
>>> selection_sort([54, 26, 93, 17, 77, 31, 44, 55, 20])
[17, 26, 93, 54, 77, 31, 44, 55, 20]
[17, 20, 93, 54, 77, 31, 44, 55, 26]
[17, 20, 26, 54, 77, 31, 44, 55, 93]
[17, 20, 26, 31, 77, 54, 44, 55, 93]
[17, 20, 26, 31, 44, 54, 77, 55, 93]
[17, 20, 26, 31, 44, 54, 77, 55, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]

>>> selection_sort(["Aisha", "Nadia", "Waqar", "Saleha", "Hasan",
"Shahid", "Shah Jamal", "Abdullah", "Umair", "Taj"])
['Abdullah', 'Nadia', 'Waqar', 'Saleha', 'Hasan', 'Shahid', 'Shah Jamal',
'Aisha', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Waqar', 'Saleha', 'Hasan', 'Shahid', 'Shah Jamal',
'Nadia', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Saleha', 'Waqar', 'Shahid', 'Shah Jamal',
'Nadia', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Waqar', 'Shahid', 'Shah Jamal',
'Saleha', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shahid', 'Shah Jamal',
'Waqar', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Waqar', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Waqar', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
```

CANDIDATE ANSWER

Language used: **Python 3**

```python

def selection_sort(lst):
    for i in range(0,len(lst)):
```

```
        minimum=i
        for j in range(i,len(lst)):
            if lst[j]<lst[minimum]:
                minimum=j
        lst[i],lst[minimum]=lst[minimum],lst[i]
            #temp=lst[i]
            #lst[i]=lst[minimum]
            #lst[minimum]=temp
        print(lst)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.062 sec	9 KB
Testcase 1	Easy	Sample case	Success	10	0.0467 sec	8.84 KB
Testcase 2	Easy	Sample case	Success	10	0.0775 sec	9.11 KB
Testcase 3	Easy	Sample case	Success	10	0.0726 sec	8.91 KB
Testcase 4	Easy	Sample case	Success	10	0.052 sec	8.96 KB
Testcase 5	Easy	Sample case	Success	10	0.0481 sec	8.8 KB
Testcase 6	Easy	Sample case	Success	10	0.065 sec	8.96 KB
Testcase 7	Easy	Sample case	Success	10	0.0474 sec	8.97 KB
Testcase 8	Easy	Sample case	Success	10	0.0825 sec	8.95 KB
Testcase 9	Easy	Sample case	Success	10	0.0821 sec	9.02 KB
Testcase 10	Easy	Sample case	Success	10	0.0495 sec	8.9 KB
Testcase 11	Easy	Sample case	Success	10	0.0427 sec	9 KB
Testcase 12	Easy	Sample case	Success	10	0.0527 sec	8.91 KB
Testcase 13	Easy	Sample case	Success	10	0.0666 sec	9 KB

No Comments

QUESTION 2

Correct Answer

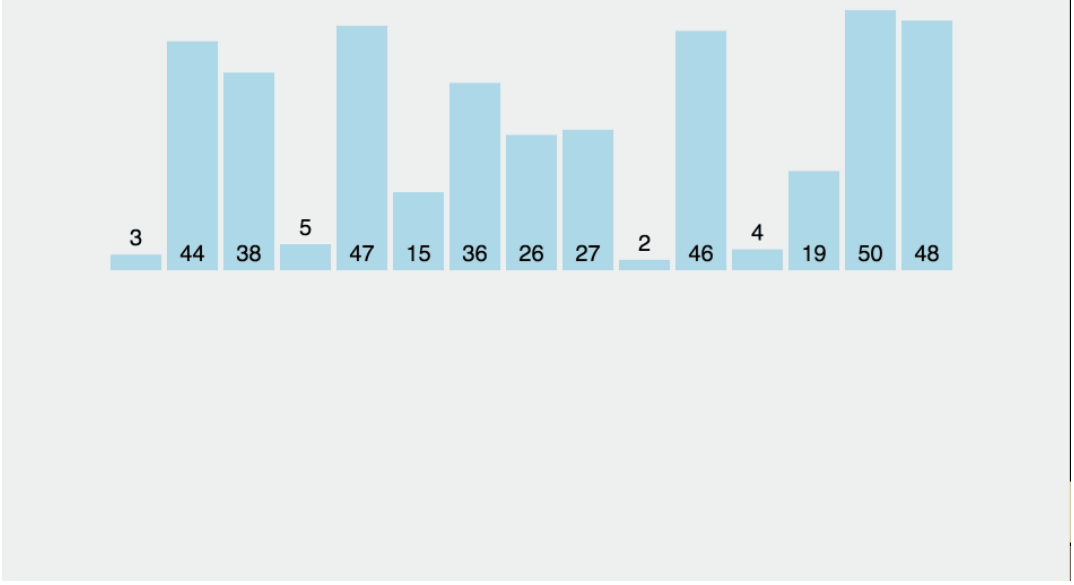
Score 130

Insertion Sort › Coding

QUESTION DESCRIPTION

Insertion sort is a simple sorting algorithm, a comparison sort in which the sorted array (or list) is built one entry at a time. An example of an insertion sort occurs in everyday life while playing cards. To sort the cards in your hand you extract a card, shift the remaining cards, and then insert the extracted card in the correct place. This process is repeated until all the cards are in the correct sequence.

Sorting Algorithm Simulation: https://visualgo.net/bn/sortin



Write a function `insertion_sort(list)` that take a `list` and sort it using **Insertion Sort Algorithm**. Print each pass or iteration.

```
>>> insertion_sort([54, 26, 93, 17, 77, 31, 44, 55, 20])
[26, 54, 93, 17, 77, 31, 44, 55, 20]
[26, 54, 93, 17, 77, 31, 44, 55, 20]
[17, 26, 54, 93, 77, 31, 44, 55, 20]
[17, 26, 54, 77, 93, 31, 44, 55, 20]
[17, 26, 31, 54, 77, 93, 44, 55, 20]
[17, 26, 31, 44, 54, 77, 93, 55, 20]
[17, 26, 31, 44, 54, 55, 77, 93, 20]
[17, 20, 26, 31, 44, 54, 55, 77, 93]

>>> insertion_sort(["Aisha", "Nadia", "Waqar", "Saleha", "Hasan",
"Shahid", "Shah Jamal", "Abdullah", "Umair", "Taj"])
['Aisha', 'Nadia', 'Waqar', 'Saleha', 'Hasan', 'Shahid', 'Shah Jamal',
'Abdullah', 'Umair', 'Taj']
['Aisha', 'Nadia', 'Waqar', 'Saleha', 'Hasan', 'Shahid', 'Shah Jamal',
'Abdullah', 'Umair', 'Taj']
['Aisha', 'Nadia', 'Saleha', 'Waqar', 'Hasan', 'Shahid', 'Shah Jamal',
'Abdullah', 'Umair', 'Taj']
['Aisha', 'Hasan', 'Nadia', 'Saleha', 'Waqar', 'Shahid', 'Shah Jamal',
'Abdullah', 'Umair', 'Taj']
['Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shahid', 'Waqar', 'Shah Jamal',
'Abdullah', 'Umair', 'Taj']
['Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid', 'Waqar',
'Abdullah', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Waqar', 'Umair', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Umair', 'Waqar', 'Taj']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
```

CANDIDATE ANSWER

Language used: **Python 3**

```python

def insertion_sort(lst):
    for i in range(1,len(lst)):
        current=lst[i]
```

```
        #free=i
        while (i>0) and (lst[i-1]>current):
            lst[i]=lst[i-1]
            i-=1
        lst[i]=current
        print(lst)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	Success	10	0.0943 sec	8.98 KB
TestCase 1	Easy	Sample case	Success	10	0.0787 sec	8.81 KB
TestCase 2	Easy	Sample case	Success	10	0.0821 sec	9.12 KB
TestCase 3	Easy	Sample case	Success	10	0.0706 sec	9.17 KB
TestCase 4	Easy	Sample case	Success	10	0.1119 sec	8.95 KB
TestCase 5	Easy	Sample case	Success	10	0.0535 sec	8.9 KB
TestCase 6	Easy	Sample case	Success	10	0.0417 sec	8.86 KB
TestCase 7	Easy	Sample case	Success	10	0.0481 sec	8.95 KB
TestCase 8	Easy	Sample case	Success	10	0.0521 sec	8.9 KB
TestCase 10	Easy	Sample case	Success	10	0.0483 sec	8.98 KB
TestCase 11	Easy	Sample case	Success	10	0.0707 sec	8.81 KB
TestCase 12	Easy	Sample case	Success	10	0.074 sec	8.77 KB
TestCase 13	Easy	Sample case	Success	10	0.0422 sec	8.81 KB

No Comments

QUESTION 3

Needs Review

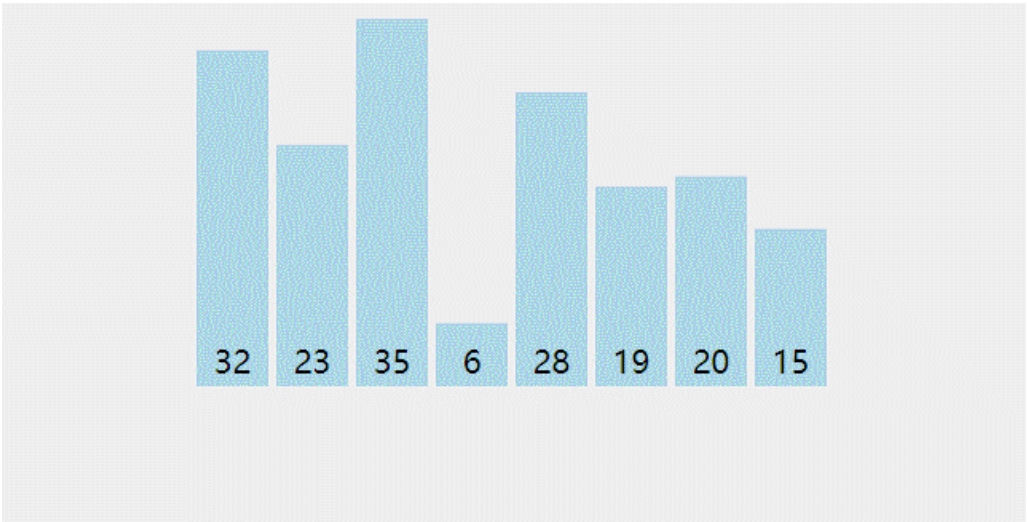
Score 140

Bubble Sort › Coding

QUESTION DESCRIPTION

The bubble sort makes multiple passes through a list. It compares adjacent items and exchanges those that are out of order. Each pass through the list places the next largest value in its proper place. In essence, each item “bubbles” up to the location where it belongs.

Sorting Algorithm Simulation: https://visualgo.net/en/sorting



Write a function **`bubble_sort(list)`** that take a `list` and sort it using **Bubble Sort Algorithm**. Print each pass or iteration.

```
>>> bubble_sort([54, 26, 93, 17, 77, 31, 44, 55, 20])
[26, 54, 17, 77, 31, 44, 55, 20, 93]
[26, 17, 54, 31, 44, 55, 20, 77, 93]
[17, 26, 31, 44, 54, 20, 55, 77, 93]
[17, 26, 31, 44, 20, 54, 55, 77, 93]
[17, 26, 31, 20, 44, 54, 55, 77, 93]
[17, 26, 20, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]
[17, 20, 26, 31, 44, 54, 55, 77, 93]

>>> bubble_sort(["Aisha", "Nadia", "Waqar", "Saleha", "Hasan", "Shahid",
"Shah Jamal", "Abdullah", "Umair", "Taj"])
['Aisha', 'Nadia', 'Saleha', 'Hasan', 'Shahid', 'Shah Jamal', 'Abdullah',
'Umair', 'Taj', 'Waqar']
['Aisha', 'Nadia', 'Hasan', 'Saleha', 'Shah Jamal', 'Abdullah', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Aisha', 'Hasan', 'Nadia', 'Saleha', 'Abdullah', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Aisha', 'Hasan', 'Nadia', 'Abdullah', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Aisha', 'Hasan', 'Abdullah', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Aisha', 'Abdullah', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
['Abdullah', 'Aisha', 'Hasan', 'Nadia', 'Saleha', 'Shah Jamal', 'Shahid',
'Taj', 'Umair', 'Waqar']
```

CANDIDATE ANSWER

Language used: **Python 3**

```python

def bubble_sort(lst):
    if len(lst)==1:
        print(lst)
        return
    for k in range(len(lst)-1):
        for i in range(len(lst)-k-1):
            if lst[i]>lst[i+1]:
                lst[i],lst[i+1]=lst[i+1],lst[i]
        print(lst)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0818 sec	9.17 KB
Testcase 1	Easy	Sample case	Success	10	0.0543 sec	8.99 KB
Testcase 2	Easy	Sample case	Success	10	0.0846 sec	8.79 KB
Testcase 3	Easy	Sample case	Success	10	0.0959 sec	9.04 KB
Testcase 4	Easy	Sample case	Success	10	0.0414 sec	8.97 KB

Testcase 5	Easy	Sample case	Success	10	0.0465 sec	9 KB
Testcase 6	Easy	Sample case	Success	10	0.051 sec	9.18 KB
Testcase 7	Easy	Sample case	Success	10	0.0904 sec	8.95 KB
Testcase 8	Easy	Sample case	Success	10	0.0462 sec	8.99 KB
Testcase 9	Easy	Sample case	Success	10	0.0667 sec	8.88 KB
Testcase 10	Easy	Sample case	Success	10	0.0763 sec	8.94 KB
Testcase 11	Easy	Sample case	Success	10	0.0746 sec	8.96 KB
Testcase 12	Easy	Sample case	Success	10	0.1047 sec	8.96 KB
Testcase 13	Easy	Sample case	Success	10	0.0754 sec	8.91 KB

No Comments

QUESTION 4

Needs Review

Score 60

Sort Matrix By Row › Coding

QUESTION DESCRIPTION

Using ***Selection Sort***, write a function `sort_matrix_by_row` that takes a `matrix` and sort the elements in the **rows** of a Matrix.

```
>>> sort_matrix_by_row([[5, 8, 1], [6, 7, 3], [5, 4, 9]])
[[1, 5, 8], [3, 6, 7], [4, 5, 9]]

>>> sort_matrix_by_row([['chair','table', 'house'],['square', 'rectangle',
'triangle'],['motor cycle', 'car', 'truck']])
[['chair', 'house', 'table'],['rectangle', 'square', 'triangle'],['car',
'motor cycle', 'truck']]
```

CANDIDATE ANSWER

Language used: **Python 3**

```python

def sort_matrix_by_row(lst):
    for k in range(len(lst)):
        inner_lst=lst[k]
        for i in range(0,len(inner_lst)):
            minimum=i
            for j in range(i,len(inner_lst)):
                if inner_lst[j]<inner_lst[minimum]:
                    minimum=j
            inner_lst[i],inner_lst[minimum]=inner_lst[minimum],inner_lst[i]
            #temp=lst[i]
            #lst[i]=lst[minimum]
            #lst[minimum]=temp
    return lst

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0533 sec	8.86 KB

Testcase 1	Easy	Sample case	Success	10	0.0458 sec	8.91 KB
Testcase 2	Easy	Sample case	Success	10	0.053 sec	8.79 KB
Testcase 3	Easy	Sample case	Success	10	0.0513 sec	8.98 KB
Testcase 4	Easy	Sample case	Success	10	0.0499 sec	9.07 KB
Testcase 5	Easy	Sample case	Success	10	0.1092 sec	8.91 KB

No Comments

QUESTION 5

Needs Review

Score 60

Sort Matrix By Column Number › Coding

QUESTION DESCRIPTION

Using **Selection Sort**, Write a function `sort_matrix_by_columnNumber` which takes as parameter a `matrix` (in form of Nested List) and a `columnNumber` and sort it by column number.

```
>>> sort_matrix_by_columnNumber([[5, 8, 1], [6, 7, 3], [5, 4, 9]], 0)
[[5, 8, 1], [5, 4, 9], [6, 7, 3]]

>>> sort_matrix_by_columnNumber([['square', 'rectangle', 'triangle'],
['chair','table', 'house'],['motor cycle', 'car', 'truck']], 2)
[['chair', 'table', 'house'], ['square', 'rectangle', 'triangle'], ['motor
cycle', 'car', 'truck']]
```

CANDIDATE ANSWER

Language used: **Python 3**

```python

def selection_sort(lst,new_lst):
    for i in range(len(lst)):
        minimum=i
        for j in range(i+1,len(lst)):
            if lst[j]<lst[minimum]:
                minimum=j
        lst[i],lst[minimum]=lst[minimum],lst[i]
        new_lst[i],new_lst[minimum]=new_lst[minimum],new_lst[i]
            #temp=lst[i]
            #lst[i]=lst[minimum]
            #lst[minimum]=temp
    return new_lst

def sort_matrix_by_columnNumber(lst,column):
    sort=[]
    for i in lst:
        sort.append(i[column])
    return(selection_sort(sort,lst))
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	20	0.1229 sec	9.03 KB
Testcase 1	Easy	Sample case	Success	20	0.1721 sec	9.12 KB

Testcase 2	Easy	Sample case	Success	20	0.0721 sec	9.04 KB

No Comments

QUESTION 6

Correct Answer

Score 40

Sorting Rectangle Records › Coding

QUESTION DESCRIPTION

We have the following rectangle records in a list of dictionaries.

```
rectangle_records = [{"ID": "Rect1", "Length": 40, "Breadth": 25, "Color": "red"}, {"ID": "Rect2", "Length": 30, "Breadth": 20, "Color": "blue"}, {"ID": "Rect3", "Length": 70, "Breadth": 45, "Color": "green"}, {"ID": "Rect4", "Length": 20, "Breadth": 10, "Color": "purple"}]
```

Using ***Insertion Sort***, Write a function `sort_rectangles` that takes `rectangle_records` and `record_title` as a parameter and sort the rectangle_records in ascending order by record_title. You need to return the updated rectangle_records list.

NOTE: The type of record_title input can be `ID`, `Length`, `Breadth`, or `Color`.

```
>>> sort_rectangles(rectangle_records, "ID")
[{"ID": "Rect1", "Length": 40, "Breadth": 25, "Color": "red"}, {"ID": "Rect2", "Length": 30, "Breadth": 20, "Color": "blue"}, {"ID": "Rect3", "Length": 70, "Breadth": 45, "Color": "green"}, {"ID": "Rect4", "Length": 20, "Breadth": 10, "Color": "purple"}]

>>> sort_rectangles(rectangle_records, "Length")
[{'ID': 'Rect4', 'Length': 20, 'Breadth': 10, 'Color': 'purple'}, {'ID': 'Rect2', 'Length': 30, 'Breadth': 20, 'Color': 'blue'}, {'ID': 'Rect1', 'Length': 40, 'Breadth': 25, 'Color': 'red'}, {'ID': 'Rect3', 'Length': 70, 'Breadth': 45, 'Color': 'green'}]

>>> sort_rectangles(rectangle_records, "Breadth")
[{'ID': 'Rect4', 'Length': 20, 'Breadth': 10, 'Color': 'purple'}, {'ID': 'Rect2', 'Length': 30, 'Breadth': 20, 'Color': 'blue'}, {'ID': 'Rect1', 'Length': 40, 'Breadth': 25, 'Color': 'red'}, {'ID': 'Rect3', 'Length': 70, 'Breadth': 45, 'Color': 'green'}]

>>> sort_rectangles(rectangle_records, "Color")
[{'ID': 'Rect2', 'Length': 30, 'Breadth': 20, 'Color': 'blue'}, {'ID': 'Rect3', 'Length': 70, 'Breadth': 45, 'Color': 'green'}, {'ID': 'Rect4', 'Length': 20, 'Breadth': 10, 'Color': 'purple'}, {'ID': 'Rect1', 'Length': 40, 'Breadth': 25, 'Color': 'red'}]
```

CANDIDATE ANSWER

Language used: **Python 3**

```python

def insertion_sort(lst,lst2):
    for i in range(1,len(lst)):
        current=lst[i]
        current2=lst2[i]
        #free=i
        while (i>0) and (lst[i-1]>current):
            lst[i]=lst[i-1]
            lst2[i]=lst2[i-1]
```

```
            i-=1
            lst[i]=current
            lst2[i]=current2
    return(lst2)
def sort_rectangles(rectangle_records,record_title):
    sort=[]
    for x in rectangle_records:
        sort.append(x.get(record_title))
    return insertion_sort(sort,rectangle_records)

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	Success	10	0.0501 sec	9.04 KB
Testcase 1	Easy	Sample case	Success	10	0.0444 sec	8.92 KB
Testcase 2	Easy	Sample case	Success	10	0.062 sec	8.83 KB
Testcase 3	Easy	Sample case	Success	10	0.0954 sec	9.01 KB

No Comments