You can view this report online at : https://www.hackerrank.com/x/tests/1541792/candidates/50828424/report

Full Name:	Breeha Qasim
Email:	bq08283@st.habib.edu.pk
Test Name:	**CS102 - Lab 9 - Spring 2023**
Taken On:	10 Mar 2023 11:56:57 PKT
Time Taken:	4367 min 3 sec/ 4320 min
Work Experience:	< 1 years
Invited by:	Aisha
Skills Score:	
Tags Score:	

100%

400/400

scored in **CS102 - Lab 9 - Spring 2023** in 4367 min 3 sec on 10 Mar 2023 11:56:57 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	Pair Of Elements having Smallest Absolute Difference > **Coding**	18 min 16 sec	70/ 70	✓
Q2	Sort An Array According To Absolute Difference With Given Value > **Coding**	25 min 6 sec	40/ 40	✓
Q3	Resize > **Coding**	3 hour 23 min 5 sec	20/ 20	✓
Q4	Delete > **Coding**	11 min 36 sec	90/ 90	!
Q5	Sum of Two Numbers Improved > **Coding**	14 min 48 sec	40/ 40	✓
Q6	Find The Missing Number > **Coding**	4 hour 58 min 59 sec	140/ 140	✓

QUESTION 1

✓ Correct Answer

Score 70

Pair Of Elements having Smallest Absolute Difference > Coding

QUESTION DESCRIPTION

Given a list of unsorted integers. Find the pair of elements that have the smallest absolute difference between them. If there are multiple pairs, find them all.

Implement a function `smallest_absdiff_pairs` that take a list of numbers and returns a list of pair of elements that have the smallest absolute difference between them.

```
>>> smallest_absdiff_pairs([5, 4, 3, 2])
[(2, 3), (3, 4), (4, 5)]

>>> smallest_absdiff_pairs([-20, -3916237, -357920, -3620601, 7374819,
-7330761, 30, 6246457, -6461594, 266854, -520, -470])
[(-520, -470) , (-20, 30)]

>>> smallest_absdiff_pairs([-20, -3916237, -357920, -3620601, 7374819,
-7330761, 30, 6246457, -6461594, 266854])
[(-20, 30)]
```

CANDIDATE ANSWER

Language used: **Python 3**

```python
import ast
lst = input()
lst = ast.literal_eval(lst)
def selectionsort(lst):
    length=len(lst)
    for x in range(0,length):
        minima=x
        for i in range(x,length):
            if lst[i]<lst[minima]:
                minima=i
        lst[x],lst[minima]=lst[minima],lst[x]

def smallest_absdiff_pairs(lst):
    empty=[]
    length2=len(lst)
    selectionsort(lst)
    for j in range(length2-1):
        empty.append((lst[j],lst[j+1]))
    empty2=[]
    for j in empty:
        empty2.append(j[1]-j[0])
    minimum=min(empty2) #finding smallest distance absolute difference
between pairs
    empty3=[]
    for j in range(len(empty2)):
        if empty2[j]==minimum:
            empty3.append(empty[j])
    return empty3


print(smallest_absdiff_pairs(lst))
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0543 sec	9 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.0743 sec	8.98 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.0829 sec	9.14 KB
Testcase 3	Easy	Sample case	✓ Success	10	0.0443 sec	8.96 KB

Testcase 4	Easy	Hidden case	✓ Success	10	0.0483 sec	9.11 KB
Testcase 5	Easy	Hidden case	✓ Success	10	0.0887 sec	10.1 KB
Testcase 6	Easy	Hidden case	✓ Success	10	2.8959 sec	20.1 KB

No Comments

QUESTION 2

✓

Correct Answer

Score 40

Sort An Array According To Absolute Difference With Given Value › Coding

QUESTION DESCRIPTION

Write a function `sort_abs_difference(lst, x)` that takes a `list` of distinct elements and a number `x`.

You have to arrange array elements according to the absolute difference with x, i.e., element having minimum difference comes first and so on. If two or more elements are at equal distance arrange them in same sequence as in the given array.

Return the sorted list.

```
>> sort_abs_difference([10, 5, 3, 9, 2], 7)
[5, 9, 10, 3, 2]

>> sort_abs_difference([1, 2, 3, 4, 5], 6)
[5, 4, 3, 2, 1]

>> sort_abs_difference([2, 6, 8, 3], 5)
[6, 3, 2, 8]
```

CANDIDATE ANSWER

Language used: **Python 3**

```python
def bubblesort(lst):
    length=len(lst)
    if length==1:
        return
    else:
        for k in range(length-1):
            for i in range(length-k-1):
                if lst[i][0]>lst[i+1][0]:
                    lst[i],lst[i+1]=lst[i+1],lst[i]
def sort_abs_difference(lst,x):
    empty=[]
    length=len(lst)
    for i in range(length):
        empty.append((abs(x-lst[i]),lst[i])) #3,10 2,5 4,3 2,9 5,2
    bubblesort(empty) #sorting all the elements
    empty2=[]
    for i in empty: #running looop on sorted list then again appending to new
list
        empty2.append(i[1])
    return empty2
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.079 sec	9.09 KB

Testcase 1	Easy	Sample case	✓ Success	10	0.1294 sec	9.11 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.0881 sec	9.05 KB
Testcase 3	Easy	Sample case	✓ Success	10	0.0876 sec	8.9 KB

No Comments

QUESTION 3

✓ Correct Answer

Score 20

Resize › Coding

QUESTION DESCRIPTION

Description

Implement the `resize()` function for hash tables. This function is called when the non None values in a hash table fill up at least 2/3 of the hash table. When resize is called, it will first creates a new hash table of 3 times the size of the original hash table and then it takes each key value pair from the original hash table, hashes the key to get a new location in the new hash table and puts the key value pair there.

Sample

Input	2 4 hello 19 world 10 1 2 3 4 hello world 1 3
Output	19 10 2 4
Explanation	The first line of the input is `size` of the hash table. The second line of the input is `n` which is the amount of entries to do in the hash table. The next `n` inputs are in separate lines in the form `key data` All remaining `n` inputs are the keys to validate if the functions work correctly Note that the size of Hashtable is initially 2 and there are more than 3 amount of entries to add in the hash table, so we would have to call resize() function when 2/3 of the hashtable is filled, otherwise it will not work as expected.

CANDIDATE ANSWER

Language used: **Python 3**

```
size = int(input())
global keys
global values
keys = [None] * size
values = [None] * size
def hashing(key):
    #key=str(key)
    suming=0
    for i in range(len(key)):
        temp=ord(str(key[i]))
        suming+=temp
    index = suming%len(keys)
    #print(index)
    return index
def rehash(keys,oldhash):
    return((oldhash+1)%len(keys))
def put(keys, values, key, data):
    counter=0
    for i in values:
        if i== None:
            counter+=1
    if counter<= (2/3)*len(values):
        keys, values = resize(keys,values)
        index = hashing(key)
        if keys[index]== None:
            keys[index]=key
            values[index]= data
            #print(keys,values)
            return keys, values
        else:
            while keys[index]!= None :
                index=rehash(keys, index)
            keys[index]=key
            values[index]=data
            #print(keys,values)
            return keys, values
    else:
        index= hashing(key)
        #print(index, len(keys))
        if keys[index]==None:
            keys[index]= key
            values[index]= data
            return keys, values
        else:

            while keys[index]!= None :
                index=rehash( keys, index)
            keys[index]=key
            values[index]=data
            return keys, values


def get(keys, values, key):   # you may add more params
    index= hashing(key)
    if keys[index]==key:
        return values[index]
    else:
        j=0
        found=False
        while j<= len(keys):
            index= rehash(keys,index)
            if keys[index]==key:
                found=True
                break
            j+=1
        if found== True:
            return values[index]
        else:
            return
def resize(keys, values):
    tempk=keys
    tempv=values
    keys=[]
    values=[]
    for i in range(len(tempk)*3):
        keys.append(None)
        values.append(None)
    for i in range(len(tempk)):
        if tempk[i]== None:
            pass
        else:
            index= hashing(tempk[i])
            if keys[index]==None:
                keys[index]=tempk[i]
                values[index]=tempv[i]
            else:
                while keys[index]!=None:
                    index= rehash(keys, index)
                keys[index]=tempk[i]
                values[index]=tempv[i]
    return keys, values

num = int(input())   # number of items to be added to the hash table
for i in range(num):
    val = input().split(" ")
    key = (val[0])
    val = (val[1])
    keys, values= put(keys, values, key, val)

for i in range(num):
    key = (input())
    print(get(keys, values, key))

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0375 sec	8.17 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.0506 sec	8.18 KB

No Comments

QUESTION 4

Needs Review

Score 90

Delete › Coding

QUESTION DESCRIPTION

Description

Create the `delete` function of hash tables. The function works similarly to get but instead of getting you the value associated to the given key it removes both the key and the value from the hash table. When deleting records from a hash table, there are two important considerations.

1. Deleting a record must not hinder later searches. In other words, the search process must still pass through the newly emptied slot to reach records whose probe sequence passed through this slot. Thus, the delete process cannot simply mark the slot as empty, because this will isolate records further down the probe sequence.
2. We do not want to make positions in the hash table unusable because of deletion. The freed slot should be available to a future insertion.

Both of these problems can be resolved by placing a special mark in place of the deleted record, called a tombstone. The tombstone indicates that a record once occupied the slot but does so no longer. If a tombstone is encountered when searching along a probe sequence, the search procedure continues with the search.

Sample

Input	10 2 hello 10 world 19 3 remove hello get hello get world
Output	Not Found 19
Explanation	The first line of the input is `size` of the hash table. The second line of the input is `n` which is the amount of entries to do in the hash table. The next `n` inputs are in separate lines in the form `key data` The next input is number of queries on hash table in the form of `operation key`. If the operation is remove, remove the key and it's corresponding value from the Hash table. If the operation is get, get the corresponding value of the key. If the key is not found, return "Not Found".

CANDIDATE ANSWER

Language used: **Python 3**

```
size = int(input())
global keys
global values
keys = [None] * size
values = [None] * size
def hashing(key):
    key=str(key)
    suming=0
    for i in range(len(key)):
        ASCII=ord(str(key[i]))
        suming+=ASCII
    index = suming%len(keys)
    return index
def rehash(keys,oldhash):
    return((oldhash+1)%len(keys))
def put(keys, values, key, data):
    counter=0
    for i in values:
        if i== None:
            counter+=1
    if counter<= (2/3)*len(values):
        keys, values = resize(keys,values)
        index = hashing(key)
        if keys[index]== None:
            keys[index]=key
            values[index]= data
            #print(keys,values)
            return keys, values
        else:
            while keys[index]!= None :
                index=rehash(keys, index)
            keys[index]=key
            values[index]=data
            #print(keys,values)
            return keys, values
    else:
        index= hashing(key)
        #print(index, len(keys))
        if keys[index]==None:
            keys[index]= key
            values[index]= data
            return keys, values
        else:
            while keys[index]!= None :
                index=rehash( keys, index)
            keys[index]=key
            values[index]=data
            #print(keys,values)
            return keys, values


def get(keys, values, key):   # you may add more params
    index= hashing(key)
    if keys[index]==key:
        return values[index]
    else:
        j=0
        found=False
        while j<= len(keys):
            index= rehash(keys,index)
            if keys[index]==key:
                found=True
                break
            j+=1
        if found== True:
            return values[index]
        else:
            return ('Not Found')

def delete(keys, values, key):   # you may add more params
    index= hashing(key)
    if keys[index]==key:
        keys[index]=None
        values[index]=None
        return keys, values
    else:
        j=0
        found=False
        while j<= len(keys):
            index= rehash(keys,index)
            if keys[index]==key:
                found=True
                break
            j+=1
        if found== True:
            keys[index]=None
            values[index]=None
            return keys, values
        else:
            return
def resize(keys, values):
    tempk=keys
    tempv=values
    keys=[]
    values=[]
    for i in range(len(tempk)*3):
        keys.append(None)
        values.append(None)
```

```python
    for i in range(len(tempk)):
        if tempk[i]== None:
            pass
        else:
            index= hashing(tempk[i])
            if keys[index]==None:
                keys[index]=tempk[i]
                values[index]=tempv[i]
            else:
                while keys[index]!=None:
                    index= rehash(keys, index)
                keys[index]=tempk[i]
                values[index]=tempv[i]
    #print(keys,values)
    return keys, values

num = int(input())   # number of items to be added to the hash table
for i in range(num):
    val = input().split(" ")
    key = (val[0])
    val = (val[1])
    keys, values= put(keys, values, key, val)

queries = int(input())
for i in range(queries):
    query = input().split(" ")
    if query[0]=="remove":
        key = (query[1])
        keys, values = delete(keys, values, key)
    elif query[0]=="get":
        key = (query[1])
        print(get(keys, values, key))

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0287 sec	8.22 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.0286 sec	8.47 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.0834 sec	8.24 KB
Testcase 3	Easy	Sample case	✓ Success	10	0.0651 sec	8.18 KB
Testcase 4	Easy	Sample case	✓ Success	10	0.0524 sec	8.23 KB
Testcase 5	Easy	Sample case	✓ Success	10	0.039 sec	8.36 KB
Testcase 6	Easy	Sample case	✓ Success	10	0.0282 sec	8.19 KB
Testcase 7	Easy	Sample case	✓ Success	10	0.0389 sec	8.27 KB
Testcase 8	Easy	Sample case	✓ Success	10	0.0338 sec	8.32 KB

No Comments

QUESTION 5

✓ Correct Answer

Score 40

Sum of Two Numbers Improved › Coding

QUESTION DESCRIPTION

Write a function `sumOfTwo` that takes as parameters *list of numbers* & the *target_sum*. The function returns the indices of the two numbers, the sum of whom is equal to the target_sum. The indices can not be of the same number. Do this in complexity of O(n).

```
>>> sumOfTwo([2,3,6,5], 5)
[0, 1]

>>> sumOfTwo([6,5,3,7,2,1,9,3,10], 19])
[6, 8]

>>> sumOfTwo([3,3], 6)
[0, 1]
```

CANDIDATE ANSWER

Language used: **Python 3**

```python
def sumOfTwo(lstnumbers,final):
    x=dict()
    length=len(lstnumbers)
    for k in range(length):
        a=lstnumbers[k]
        if a in x:
            return [x[a],k]
        else:
            z=final-a
            x[z]=k
    return None

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0491 sec	8.95 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.0632 sec	9.06 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.0563 sec	8.97 KB
Testcase 3	Easy	Sample case	✓ Success	10	0.0584 sec	9.02 KB

No Comments

QUESTION 6

✓ Correct Answer

Score 140

Find The Missing Number › Coding

QUESTION DESCRIPTION

You are given a list of n-1 integers such that each integer is equal to some powers of 2 and these integers are in the range of 1 to 2^(n-1).
There are no duplicates in the list and the integers are in ascending order.
One of the integers is missing in the list.

Implement the function ***findMissingNumber*** that takes a list of Powers of 2 numbers and the size and returns the missing integer in O(log n) time.

Testcases Contribution Credit: Muhammad Haroon Khan (mk03985)

```
>> findMissingNumber([1, 2, 4, 8], 5)
16

>> findMissingNumber([2, 4, 8], 4)
```

```
1

>> findMissingNumber([1, 2, 4, 16, 32, 64], 7)
8
```

CANDIDATE ANSWER

Language used: **Python 3**

```python
def findMissingNumber(powerTwoList, size):
    minima=0
    maxima=size - 2
    while minima<=maxima:
        mid=(maxima+minima)//2
        if powerTwoList[mid]!=2**mid:
            maxima=mid-1
        else:
            minima=mid+1
    return 2**minima

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✓ Success	10	0.0796 sec	8.98 KB
Testcase 1	Easy	Sample case	✓ Success	10	0.0468 sec	9.15 KB
Testcase 2	Easy	Sample case	✓ Success	10	0.0772 sec	8.98 KB
Testcase 3	Easy	Sample case	✓ Success	10	0.0613 sec	9.04 KB
Testcase 4	Easy	Sample case	✓ Success	10	0.0426 sec	9.02 KB
Testcase 5	Easy	Sample case	✓ Success	10	0.0458 sec	9.04 KB
Testcase 6	Easy	Sample case	✓ Success	10	0.0471 sec	8.94 KB
Testcase 7	Easy	Sample case	✓ Success	10	0.0525 sec	9.22 KB
Testcase 8	Easy	Sample case	✓ Success	10	0.0845 sec	9.18 KB
Testcase 9	Easy	Sample case	✓ Success	10	0.1001 sec	9.28 KB
Testcase 10	Easy	Sample case	✓ Success	10	0.0435 sec	9.04 KB
Testcase 11	Easy	Sample case	✓ Success	10	0.0566 sec	9.11 KB
Testcase 12	Easy	Sample case	✓ Success	10	0.0548 sec	9.06 KB
Testcase 13	Easy	Sample case	✓ Success	10	0.0975 sec	9.01 KB

No Comments