Rajalakshmi Engineering College

Name: Krithika Raghavan
Email: 240701278@rajalakshmi.edu.in
Roll no: 240701278
Phone: 7305916492
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 15

Section 1 : MCQ

1. What is the worst-case time complexity for inserting an element in a hash table with linear probing?

Answer

O(1)

Status : Wrong ***Marks : 0/1***

2. What is the initial position for a key k in a linear probing hash table?

Answer

k % table_size

Status : Correct ***Marks : 1/1***

3. What would be the result of folding 123456 into three parts and summing: (12 + 34 + 56)?

Answer

102

Status : Correct ***Marks : 1/1***

4. Which C statement is correct for finding the next index in linear probing?

Answer

index = (index + 1) % size;

Status : Correct ***Marks : 1/1***

5. What is the output of the mid-square method for a key k = 123 if the hash table size is 10 and you extract the middle two digits of k * k?

Answer

2

Status : Wrong ***Marks : 0/1***

6. Which of these hashing methods may result in more uniform distribution with small keys?

Answer

Mid-Square

Status : Correct ***Marks : 1/1***

7. What is the primary disadvantage of linear probing?

Answer

Clustering

Status : Correct ***Marks : 1/1***

8. Which folding method divides the key into equal parts, reverses some of them, and then adds all parts?

Answer

Folding boundary method

Status : Wrong ***Marks : 0/1***

9. In C, how do you calculate the mid-square hash index for a key k, assuming we extract two middle digits and the table size is 100?

Answer

((k * k) / 10) % 100

Status : Wrong ***Marks : 0/1***

10. In the division method of hashing, the hash function is typically written as:

Answer

h(k) = k % m

Status : Correct ***Marks : 1/1***

11. Which of the following statements is TRUE regarding the folding method?

Answer

It divides the key into parts and adds them.

Status : Correct ***Marks : 1/1***

12. Which situation causes clustering in linear probing?

Answer

Poor hash function

Status : Wrong ***Marks : 0/1***

13. Which data structure is primarily used in linear probing?

Answer

Array

Status : Correct ***Marks : 1/1***

14. In division method, if key = 125 and m = 13, what is the hash index?

Answer

8

Status : Correct ***Marks : 1/1***

15. What does a deleted slot in linear probing typically contain?

Answer

A special "deleted" marker

Status : Correct ***Marks : 1/1***

16. Which of the following best describes linear probing in hashing?

Answer

Resolving collisions by linearly searching for the next free slot

Status : Correct ***Marks : 1/1***

17. Which of the following values of 'm' is recommended for the division method in hashing?

Answer

A prime number

Status : Correct ***Marks : 1/1***

18. In linear probing, if a collision occurs at index i, what is the next index checked?

Answer

(i + 1) % table_size

Status : Correct ***Marks : 1/1***

19. In the folding method, what is the primary reason for reversing alternate parts before addition?

Answer

To reduce the chance of collisions caused by similar digit patterns

Status : Correct ***Marks : 1/1***

20. What happens if we do not use modular arithmetic in linear probing?

Answer

Index goes out of bounds

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Krithika Raghavan
Email: 240701278@rajalakshmi.edu.in
Roll no: 240701278
Phone: 7305916492
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is building a basic hash table to manage student roll numbers for quick lookup. He decides to use Linear Probing to handle collisions.

Implement a hash table using linear probing where:

The hash function is: index = roll_number % table_sizeOn collision, check subsequent indexes (i+1, i+2, ...) until an empty slot is found.

You need to:

Insert a list of n student roll numbers into the hash table.Print the final state of the hash table. If a slot is empty, print -1.

Input Format

The first line of the input contains two integers n and table_size, where n is the

number of roll numbers to be inserted, and table_size is the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert into the hash table.

Output Format

The output should print a single line with table_size space-separated integers representing the final state of the hash table after all insertions.

If any slot remains unoccupied, it should be represented as -1.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4 7
50 700 76 85

Output: 700 50 85 -1 -1 -1 76

Answer

```
#include <stdio.h>

#define MAX 100

// You are using GCC
void initializeTable(int table[], int size) {
    for(int i=0;i<size;i++){
        table[i]=-1;
    }
}

int linearProbe(int table[], int size, int num) {
    int index=num%size;
    int start=index;
    while(table[index]!=-1){
        index=(index+1)%size;
        if(index==start){
            return -1;
        }
```

```
    }
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    for(int i=0;i<n;i++){
        int index=linearProbe(table,size,arr[i]);
        if(index!=-1){
            table[index]=arr[i];
        }
    }
}

void printTable(int table[], int size) {
    for(int i=0;i<size;i++){
        printf("%d",table[i]);
        if(1!=size-1){
            printf(" ");
        }
    }
}

int main() {
    int n, table_size;
    scanf("%d %d", &n, &table_size);

    int arr[MAX];
    int table[MAX];

    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);
    printTable(table, table_size);

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Krithika Raghavan
Email: 240701278@rajalakshmi.edu.in
Roll no: 240701278
Phone: 7305916492
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 2

Attempt : 2
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Priya is developing a simple student management system. She wants to store roll numbers in a hash table using Linear Probing, and later search for specific roll numbers to check if they exist.

Implement a hash table using linear probing with the following operations:

Insert all roll numbers into the hash table.For a list of query roll numbers, print "Value x: Found" or "Value x: Not Found" depending on whether it exists in the table.

Input Format

The first line contains two integers, n and table_size — the number of roll numbers to insert and the size of the hash table.

The second line contains n space-separated integers — the roll numbers to insert.

The third line contains an integer q — the number of queries.

The fourth line contains q space-separated integers — the roll numbers to search for.

Output Format

The output print q lines — for each query value x, print: "Value x: Found" or "Value x: Not Found"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5 10
21 31 41 51 61
3
31 60 51

Output: Value 31: Found
Value 60: Not Found
Value 51: Found

Answer

```
#include <stdio.h>

#define MAX 100


// You are using GCC
void initializeTable(int table[], int size) {
    for(int i=0;i<size;i++){
        table[i]=-1;
    }
}

int linearProbe(int table[], int size, int num) {
    int index=num%size;
    int start=index;
```

```
    while(table[index]!=-1){
        index=(index)+1%size;
        if(index==start){
            return -1;
        }
    }
    return index;
}

void insertIntoHashTable(int table[], int size, int arr[], int n) {
    for(int i=0;i<n;i++){
        int index=linearProbe(table,size,arr[i]);
        if(index!=-1){
            table[index]=arr[i];
        }
    }
}

int searchInHashTable(int table[], int size, int num) {
    int index=num%size;
    int start=index;
    while(table[index]!=-1){
        if(table[index]==num){
            return 1;
        }
        index=(index+1)%size;
        if(index==start){
            break;
        }
    }
    return 0;
}

int main() {
    int n, table_size;
    scanf("%d %d", &n, &table_size);

    int arr[MAX], table[MAX];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    initializeTable(table, table_size);
    insertIntoHashTable(table, table_size, arr, n);
```

```
    int q, x;
    scanf("%d", &q);
    for (int i = 0; i < q; i++) {
        scanf("%d", &x);
        if (searchInHashTable(table, table_size, x))
            printf("Value %d: Found\n", x);
        else
            printf("Value %d: Not Found\n", x);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Krithika Raghavan
Email: 240701278@rajalakshmi.edu.in
Roll no: 240701278
Phone: 7305916492
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a messaging application, users maintain a contact list with names and corresponding phone numbers. Develop a program to manage this contact list using a dictionary implemented with hashing.

The program allows users to add contacts, delete contacts, and check if a specific contact exists. Additionally, it provides an option to print the contact list in the order of insertion.

Input Format

The first line consists of an integer n, representing the number of contact pairs to be inserted.

Each of the next n lines consists of two strings separated by a space: the name of the contact (key) and the corresponding phone number (value).

The last line contains a string k, representing the contact to be checked or removed.

Output Format

If the given contact exists in the dictionary:

1. The first line prints "The given key is removed!" after removing it.
2. The next n - 1 lines print the updated contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

If the given contact does not exist in the dictionary:

1. The first line prints "The given key is not found!".
2. The next n lines print the original contact list in the format: "Key: X; Value: Y" where X represents the contact's name and Y represents the phone number.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 3
Alice 1234567890
Bob 9876543210
Charlie 4567890123
Bob

Output: The given key is removed!
Key: Alice; Value: 1234567890
Key: Charlie; Value: 4567890123

Answer

```
// You are using GCC
#include <stdio.h>
#include <string.h>

#define MAX_CONTACTS 50
#define MAX_NAME_LEN 11
```

```
#define MAX_PHONE_LEN 20

typedef struct {
    char name[MAX_NAME_LEN];
    char phone[MAX_PHONE_LEN];
} Contact;

int main() {
    int n;
    scanf("%d", &n);

    Contact contacts[MAX_CONTACTS];

    for (int i = 0; i < n; i++) {
        scanf("%s %s", contacts[i].name, contacts[i].phone);
    }

    char key[MAX_NAME_LEN];
    scanf("%s", key);

    int found = 0, index = -1;

    // Search for key
    for (int i = 0; i < n; i++) {
        if (strcmp(contacts[i].name, key) == 0) {
            found = 1;
            index = i;
            break;
        }
    }

    if (found) {
        printf("The given key is removed!");
        // Shift remaining elements to remove contact
        for (int i = index; i < n - 1; i++) {
            contacts[i] = contacts[i + 1];
        }
        n--; // One contact removed
    } else {
        printf("The given key is not found!");
    }
```

```
    // Print remaining contacts
    for (int i = 0; i < n; i++) {
        printf(" Key: %s; Value: %s", contacts[i].name, contacts[i].phone);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Krithika Raghavan
Email: 240701278@rajalakshmi.edu.in
Roll no: 240701278
Phone: 7305916492
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Develop a program using hashing to manage a fruit contest where each fruit is assigned a unique name and a corresponding score. The program should allow the organizer to input the number of fruits and their names with scores.

Then, it should enable them to check if a specific fruit, identified by its name, is part of the contest. If the fruit is registered, the program should display its score; otherwise, it should indicate that it is not included in the contest.

Input Format

The first line consists of an integer N, representing the number of fruits in the contest.

The following N lines contain a string K and an integer V, separated by a space, representing the name and score of each fruit in the contest.

The last line consists of a string T, representing the name of the fruit to search for.

Output Format

If T exists in the dictionary, print "Key "T" exists in the dictionary.".

If T does not exist in the dictionary, print "Key "T" does not exist in the dictionary.".

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 2
banana 2
apple 1
Banana
Output: Key "Banana" does not exist in the dictionary.

Answer

```
#include <stdio.h>
#include <string.h>

#define TABLE_SIZE 20
#define MAX_NAME_LEN 21

typedef struct {
    char name[MAX_NAME_LEN];
    int score;
    int is_occupied; // 0 = empty, 1 = occupied
} Fruit;

Fruit hashTable[TABLE_SIZE];

int hash(char *key) {
    int hashValue = 0;
```

```
    for (int i = 0; key[i] != '\0'; i++) {
        hashValue += key[i];
    }
    return hashValue % TABLE_SIZE;
}

void initializeTable() {
    for (int i = 0; i < TABLE_SIZE; i++) {
        hashTable[i].is_occupied = 0;
    }
}

void insert(char *name, int score) {
    int index = hash(name);
    int start = index;

    while (hashTable[index].is_occupied == 1) {
        index = (index + 1) % TABLE_SIZE;
        if (index == start) return; // Table full (should not happen with N ≤ 15)
    }

    strcpy(hashTable[index].name, name);
    hashTable[index].score = score;
    hashTable[index].is_occupied = 1;
}

int search(char *name) {
    int index = hash(name);
    int start = index;

    while (hashTable[index].is_occupied != 0) {
        if (hashTable[index].is_occupied == 1 &&
            strcmp(hashTable[index].name, name) == 0) {
            return 1; // Found
        }
        index = (index + 1) % TABLE_SIZE;
        if (index == start) break;
    }
    return 0; // Not found
}

int main() {
```

```
    int N;
    scanf("%d", &N);

    initializeTable();

    char name[MAX_NAME_LEN];
    int score;

    for (int i = 0; i < N; i++) {
        scanf("%s %d", name, &score);
        insert(name, score);
    }

    char target[MAX_NAME_LEN];
    scanf("%s", target);

    if (search(target)) {
        printf("Key \"%s\" exists in the dictionary.", target);
    } else {
        printf("Key \"%s\" does not exist in the dictionary.", target);
    }

    return 0;
}
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Krithika Raghavan
Email: 240701278@rajalakshmi.edu.in
Roll no: 240701278
Phone: 7305916492
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 7_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are provided with a collection of numbers, each represented by an array of integers. However, there's a unique scenario: within this array, one element occurs an odd number of times, while all other elements occur an even number of times. Your objective is to identify and return the element that occurs an odd number of times in this arrangement.

Utilize mid-square hashing by squaring elements and extracting middle digits for hash codes. Implement a hash table for efficient integer occurrence tracking.

Note: Hash function: squared = key * key.

Example

Input:

7

2 2 3 3 4 4 5

Output:

5

Explanation

The hash function and the calculated hash indices for each element are as follows:

2 -> hash(2*2) % 100 = 4

3 -> hash(3*3) % 100 = 9

4 -> hash(4*4) % 100 = 16

5 -> hash(5*5) % 100 = 25

The hash table records the occurrence of each element's hash index:

Index 4: 2 occurrences

Index 9: 2 occurrences

Index 16: 2 occurrences

Index 25: 1 occurrence

Among the elements, the integer 5 occurs an odd number of times (1 occurrence) and satisfies the condition of the problem. Therefore, the program outputs 5.

Input Format

The first line of input consists of an integer N, representing the size of the array.

The second line consists of N space-separated integers, representing the elements of the array.

Output Format

The output prints a single integer representing the element that occurs an odd

number of times.

If no such element exists, print -1.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 7
2 2 3 3 4 4 5

Output: 5

Answer

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define MAX_SIZE 100

// You are using GCC
unsigned int hash(int key, int tableSize) {
    int squared = key * key;
    return squared % tableSize;
}
int getOddOccurrence(int arr[], int size) {
    int table[MAX_SIZE] = {0};
    for (int i = 0; i < size; i++) {
        int idx = hash(arr[i], MAX_SIZE);
        table[idx]++;
    }
    for (int i = 0; i < size; i++) {
        int idx = hash(arr[i], MAX_SIZE);
        if (table[idx] % 2 != 0)
            return arr[i];
    }
    return -1;
}

int main() {
    int n;
```

```
    scanf("%d", &n);

    int arr[MAX_SIZE];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("%d\n", getOddOccurrence(arr, n));

    return 0;
}
```

Status : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Krithika Raghavan
Email: 240701278@rajalakshmi.edu.in
Roll no: 240701278
Phone: 7305916492
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 7_CY

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Arjun is monitoring hourly temperature data recorded continuously for multiple days. He needs to calculate the average temperature for each day based on 24 hourly readings.

Help him to implement the task using the numpy package.

Formula:

Reshape the temperature readings into rows where each row has 24 readings (one day).

Average temperature per day = mean of 24 hourly readings in each row.

Input Format

The first line of input consists of an integer value, n, representing the total number of temperature readings.

The second line of input consists of n floating-point values separated by spaces, representing hourly temperature readings.

Output Format

The output prints: avg_per_day

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 30
30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0 30.0
Output: [30.]

Answer

```
# You are using Python
import numpy as np
n = int(input())
readings = list(map(float, input().split()))
data = np.array(readings)
daily_data = data.reshape(-1, 24)
avg_per_day = np.mean(daily_data, axis=1)
print(avg_per_day)
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Rekha is a meteorologist analyzing rainfall data collected over 5 years, with monthly rainfall recorded for each year. She wants to find the total rainfall each year and also identify the month with the maximum rainfall for every year.

Help her to implement the task using the numpy package.

Formula:

Yearly total rainfall = sum of all 12 months' rainfall for each year

Month with max rainfall = index of the maximum rainfall value within the 12 months for each year (0-based index)

Input Format

The input consists of 5 lines.

Each line contains 12 floating-point values separated by spaces, representing the rainfall data (in mm) for each month of that year.

Output Format

The first line of output prints: yearly_totals

The second line of output prints: max_rainfall_months

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1.0 2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0
2.0 3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0
3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0
4.0 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0
5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0 15.0 16.0
Output: [78. 90. 102. 114. 126.]
[11 11 11 11 11]

Answer

```
# You are using Python
import numpy as np
data = []
for _ in range(5):
    row = list(map(float, input().split()))
    data.append(row)
rainfall = np.array(data)
yearly_totals = np.sum(rainfall, axis=1)
```

```
max_rainfall_months = np.argmax(rainfall, axis=1)
print(yearly_totals)
print(max_rainfall_months)
```

Status : Correct **Marks : 10/10**

3. Problem Statement

Rekha works as an e-commerce data analyst. She receives transaction data containing purchase dates and needs to extract the month and day from these dates using the pandas package.

Help her implement this task by performing the following steps:

Convert the Purchase Date column to datetime format, treating invalid date entries as NaT (missing).

Create two new columns:

Purchase Month, containing the month (as an integer) extracted from the Purchase Date.

Purchase Day, containing the day (as an integer) extracted from the Purchase Date. Keep the rest of the data as is.

Input Format

The first line of input contains an integer n, representing the number of records.

The second line contains the CSV header — comma-separated column names.

The next n lines each contain a transaction record in comma-separated format.

Output Format

The first line of output is the text:

Transformed E-commerce Transaction Data:

The next lines print the pandas DataFrame with:

The original columns (including Purchase Date, which is now in datetime format or NaT if invalid).

Two additional columns: Purchase Month and Purchase Day.

The output uses the default pandas DataFrame string representation as produced by print(transformed_df).

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
Customer,Purchase Date
Alice,2023-05-15
Bob,2023-06-20
Charlie,2023-07-01

Output: Transformed E-commerce Transaction Data:

	Customer	Purchase Date	Purchase Month	Purchase Day
0	Alice	2023-05-15	5	15
1	Bob	2023-06-20	6	20
2	Charlie	2023-07-01	7	1

Answer

```
# You are using Python
import pandas as pd
import sys
n = int(input())
header = input().split(',')
records = [input().split(',') for _ in range(n)]
df = pd.DataFrame(records, columns=header)
df["Purchase Date"] = pd.to_datetime(df["Purchase Date"], errors='coerce')
df["Purchase Month"] = df["Purchase Date"].dt.month
df["Purchase Day"] = df["Purchase Date"].dt.day
print("Transformed E-commerce Transaction Data:")
print(df)
```

Status : Correct ***Marks : 10/10***

4. Problem Statement

You are working as a data analyst for a small retail store that wants to track the stock levels of its products. Each product has a unique Name (such as "Toothpaste", "Shampoo", "Soap") and an associated Quantity in stock. Management wants to identify which products have zero stock so they can be restocked.

Write a Python program using the pandas library to help with this task. The program should:

Read the number of products, n.Read n lines, each containing the Name of the product and its Quantity, separated by a space.Convert this data into a pandas DataFrame.Identify and display the Name and Quantity of products with zero stock.If no products have zero stock, display: No products with zero stock.

Input Format

The first line contains an integer n, the number of products.

The next n lines each contain:

<Product_ID> <Quantity>

where <Product_ID> is a single word (e.g., "Shampoo") and <Quantity> is a non-negative integer (e.g., 5).

Output Format

The first line of output prints:

Products with Zero Stock:

If there are any products with zero stock, the following lines print the pandas DataFrame showing those products with two columns: Product_ID and Quantity.

The column headers Product_ID and Quantity are printed in the second line.

Each subsequent line shows the product's name and quantity, aligned under the respective headers, with no index column.

The output formatting (spacing and alignment) follows the default pandas to_string(index=False) style.

If no products have zero stock, print:

No products with zero stock.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
P101 10
P102 0
P103 5
Output: Products with Zero Stock:
Product_ID Quantity
     P102   0

Answer

```
# You are using Python
import pandas as pd
n = int(input())
data = []
for _ in range(n):
    name, qty = input().split()
    data.append([name, int(qty)])
df = pd.DataFrame(data, columns=["Product_ID", "Quantity"])
zero_stock = df[df["Quantity"] == 0]
print("Products with Zero Stock:")
```

```
if zero_stock.empty:
    print("No products with zero stock.")
else:
    print(zero_stock.to_string(index=False))
```

Status : Correct ***Marks : 10/10***

5. Problem Statement

Arjun is developing a system to monitor environmental sensors installed in different rooms of a smart building. Each sensor records multiple temperature readings throughout the day. To compare sensor data fairly despite differing scales, Arjun needs to normalize each sensor's readings so that they have a mean of zero and standard deviation of one.

Help him implement this normalization using numpy.

Normalization Formula:

Input Format

The first line of input consists of two integers: sensors (number of sensors) and samples (number of readings per sensor).

The next sensors lines each contain samples space-separated floats representing the sensor readings.

Output Format

The first line of output prints: "Normalized Sensor Data:"

The next lines print the normalized readings as a numpy array, where each row corresponds to a sensor's normalized values.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3 3
1.0 2.0 3.0
4.0 5.0 6.0
7.0 8.0 9.0

Output: Normalized Sensor Data:
[[-1.22474487 0.   1.22474487]
 [-1.22474487 0.   1.22474487]
 [-1.22474487 0.   1.22474487]]

Answer

```
# You are using Python
import numpy as np
sensors, samples = map(int, input().split())
data = []
for _ in range(sensors):
    readings = list(map(float, input().split()))
    data.append(readings)
arr = np.array(data)
normalized = (arr - arr.mean(axis=1, keepdims=True)) / arr.std(axis=1,
keepdims=True)
print("Normalized Sensor Data:")
print(normalized)
```

Status : Correct ***Marks : 10/10***

Rajalakshmi Engineering College

Name: Krithika Raghavan
Email: 240701278@rajalakshmi.edu.in
Roll no: 240701278
Phone: 7305916492
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23221_Python Programming

REC_Python_Week 7_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 46

Section 1 : Coding

1. Problem Statement

You're analyzing the daily returns of a set of financial assets over a period of time. Each day is represented as a row in a 2D array, where each column represents the return of a specific asset on that day.

Your task is to identify which days had all positive returns across every asset using numpy, and output a boolean array indicating these days.

Input Format

The first line of input consists of two integer values, rows and cols, separated by a space.

Each of the next rows lines consists of cols float values representing the returns of the assets for that day.

Output Format

The first line of output prints: "Days where all asset returns were positive:"

The second line of output prints: the boolean array positive_days, indicating True for days where all asset returns were positive and False otherwise.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3 4
0.01 0.02 0.03 0.04
0.05 0.06 0.07 0.08
-0.01 0.02 0.03 0.04
Output: Days where all asset returns were positive:
[True True False]

Answer

```
# You are using Python
import numpy as np
rows, cols = map(int, input().split())
data = [list(map(float, input().split())) for _ in range(rows)]
arr = np.array(data)
positive_days = np.all(arr > 0, axis=1)
print("Days where all asset returns were positive:")
print(positive_days)
```

Status : Correct ***Marks : 10/10***

2. Problem Statement

Arjun is a data scientist working on an image processing task. He needs to normalize the pixel values of a grayscale image matrix to scale between 0 and 1. The input image data is provided as a matrix of integers.

Help him to implement the task using the numpy package.

Formula:

To normalize each pixel value in the image matrix:

normalized_pixel = (pixel - min_pixel) / (max_pixel - min_pixel)

where min_pixel and max_pixel are the minimum and maximum pixel values in the image matrix, respectively. If all pixel values are the same, the normalized image matrix should be filled with zeros.

Input Format

The first line of input consists of an integer value, rows, representing the number of rows in the image matrix.

The second line of input consists of an integer value, cols, representing the number of columns in the image matrix.

The next rows lines each consist of cols integer values separated by a space, representing the pixel values of the image matrix.

Output Format

The output prints: normalized_image

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
3
1 2 3
4 5 6
Output: [[0. 0.2 0.4]
 [0.6 0.8 1.]]

Answer

```
# You are using Python
import numpy as np
rows = int(input())
cols = int(input())
data = [list(map(int, input().split())) for _ in range(rows)]
img = np.array(data)
min_val = np.min(img)
```

```
max_val = np.max(img)
if min_val == max_val:
    norm_img = np.zeros_like(img, dtype=float)
else:
    norm_img = (img - min_val) / (max_val - min_val)
print(norm_img)
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Arjun manages a busy customer service center and wants to analyze the distribution of customer wait times to improve service efficiency. He decides to group the wait times into intervals of 5 minutes each and count how many customers fall into each interval bucket.

Help him implement this bucketing and counting task using NumPy.

Bucketing Logic:

Divide the wait times into intervals (buckets) of size 5 minutes, e.g.:

[0–5), [5–10), [10–15), …

Use NumPy's digitize function to determine which bucket each wait time falls into.

Count the number of wait times in each bucket and generate bucket labels.

Input Format

The first line contains an integer n, the number of customer wait times recorded.

The second line contains n space-separated floating-point numbers representing the wait times (in minutes).

Output Format

The first line of output is the text:

Wait Time Buckets and Counts:

Each subsequent line prints the bucket range and the number of wait times in that bucket, formatted as:

<bucket_range>: <count>

where <bucket_range> is the lower and upper bound of the bucket (inclusive lower bound, exclusive upper bound), for example:

0-5: 3

5-10: 2

10-15: 1

The output uses the default string formatting of Python's print() function (no extra spaces, no special formatting beyond the specified lines).

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 10
2.0 3.0 7.0 8.0 12.0 14.0 18.0 19.0 21.0 25.0
Output: Wait Time Buckets and Counts:
0-5: 2
5-10: 2
10-15: 2
15-20: 2
20-25: 1

Answer

```
# You are using Python
import numpy as np
import math
n = int(input())
wait_times = np.array(list(map(float, input().split())))
max_time = np.max(wait_times)
upper = math.ceil(max_time / 5) * 5
```

```
bins = np.arange(0, upper + 1, 5)
bucket_indices = np.digitize(wait_times, bins, right=False)
counts = np.zeros(len(bins) - 1, dtype=int)
for idx in bucket_indices:
    if 1 <= idx <= len(counts):
        counts[idx - 1] += 1
print("Wait Time Buckets and Counts:")
for i in range(len(counts)):
    if counts[i] > 0:
        print(f"{int(bins[i])}-{int(bins[i+1])}: {counts[i]}")
```

Status : Partially correct ***Marks : 6/10***

4. Problem Statement

A software development company wants to classify its employees based on their years of service at the company. They want to categorize employees into three experience levels: Junior (less than 3 years), Mid (3 to 6 years, inclusive), and Senior (more than 6 years).

Experience Level Classification:

Junior: Years at Company < 3

Mid: 3 ≤ Years at Company < 6

Senior: Years at Company > 5

You need to create a Python program using the pandas library that reads employee data, processes it into a DataFrame, and adds a new column "Experience Level" to display the appropriate classification for each employee.

Input Format

First line: an integer n representing the number of employees.

Next n lines: each line has a string Name and a floating-point number Years at Company (space-separated).

Output Format

First line: "Employee Data with Experience Level:"

The employee data table printed with no index column, and with columns: Name, Years at Company, Experience Level.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
Alice 2
Bob 4
Charlie 7
Diana 3
Evan 6

Output: Employee Data with Experience Level:

Name	Years at Company	Experience Level
Alice	2.0	Junior
Bob	4.0	Mid
Charlie	7.0	Senior
Diana	3.0	Mid
Evan	6.0	Senior

Answer

```
# You are using Python
import pandas as pd
n = int(input())
names = []
years = []
for _ in range(n):
    name, y = input().split()
    names.append(name)
    years.append(float(y))
df = pd.DataFrame({"Name": names,"Years at Company": years})
def classify(years):
    if years < 3:
        return "Junior"
    elif years <6:
        return "Mid"
    else:
        return "Senior"
```

```
df["Experience Level"] = df["Years at Company"].apply(classify)
print("Employee Data with Experience Level:")
print(df.to_string(index=False))
```

Status : Correct **Marks : 10/10**

5. Problem Statement

A company conducted a customer satisfaction survey where each respondent provides their RespondentID and an optional textual Feedback. Sometimes, respondents submit their ID without any feedback or with empty feedback.

Your task is to process the survey responses using pandas to replace any missing or empty feedback with the phrase "No Response". Finally, print the cleaned survey responses exactly as shown in the sample output.

Input Format

The first line contains an integer n, the number of survey responses.

Each of the next n lines contains:

A RespondentID (a single alphanumeric string without spaces),

Followed optionally by a Feedback string, which may be empty or missing.

If no feedback is provided after the RespondentID, treat it as missing.

Output Format

Print the line:

Survey Responses with Missing Feedback Filled:

Then print the cleaned survey data as a table with two columns: RespondentID and Feedback.

The table should have the headers exactly as:

RespondentID Feedback

Print each respondent's data on a new line, aligned to match the output produced by pandas.DataFrame.to_string(index=False).

For any missing or empty feedback, print "No Response" in the Feedback column.

Maintain the spacing and alignment exactly as shown in the sample outputs.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4
101 Great service
102
103 Loved it
104

Output: Survey Responses with Missing Feedback Filled:
RespondentID Feedback
101 Great service
102 No Response
103 Loved it
104 No Response

Answer

```
# You are using Python
import pandas as pd
n = int(input())
ids = []
feedbacks = []
for _ in range(n):
    parts = input().strip().split(maxsplit=1)
    ids.append(parts[0])
    if len(parts) == 2 and parts[1].strip():
```

```
        feedbacks.append(parts[1])
    else:
        feedbacks.append("No Response")
df = pd.DataFrame({"RespondentID": ids,"Feedback": feedbacks})
print("Survey Responses with Missing Feedback Filled:")
print(df.to_string(index=False))
```

Status : Correct ***Marks : 10/10***