DBMS Project

On

Online Coding Platform Database

Done by

Thallada Pavan Siddartha

Problem Statement :

This database that we have constructed in our project basically gives a broad overview about how a Coding site or coding platform works, in general there are many coding sites that we use daily, we have analysed how the data is managed in those sites and have constructed an efficient DB.

This database consists of only essential components (entities) that covers almost all features of a efficient coding site,giving an edge to

- Participate in contests that occurs weekly,
- Announcing the results,
- Practising problems, and
- Even the Store that helps them to buy the necessary goodies.

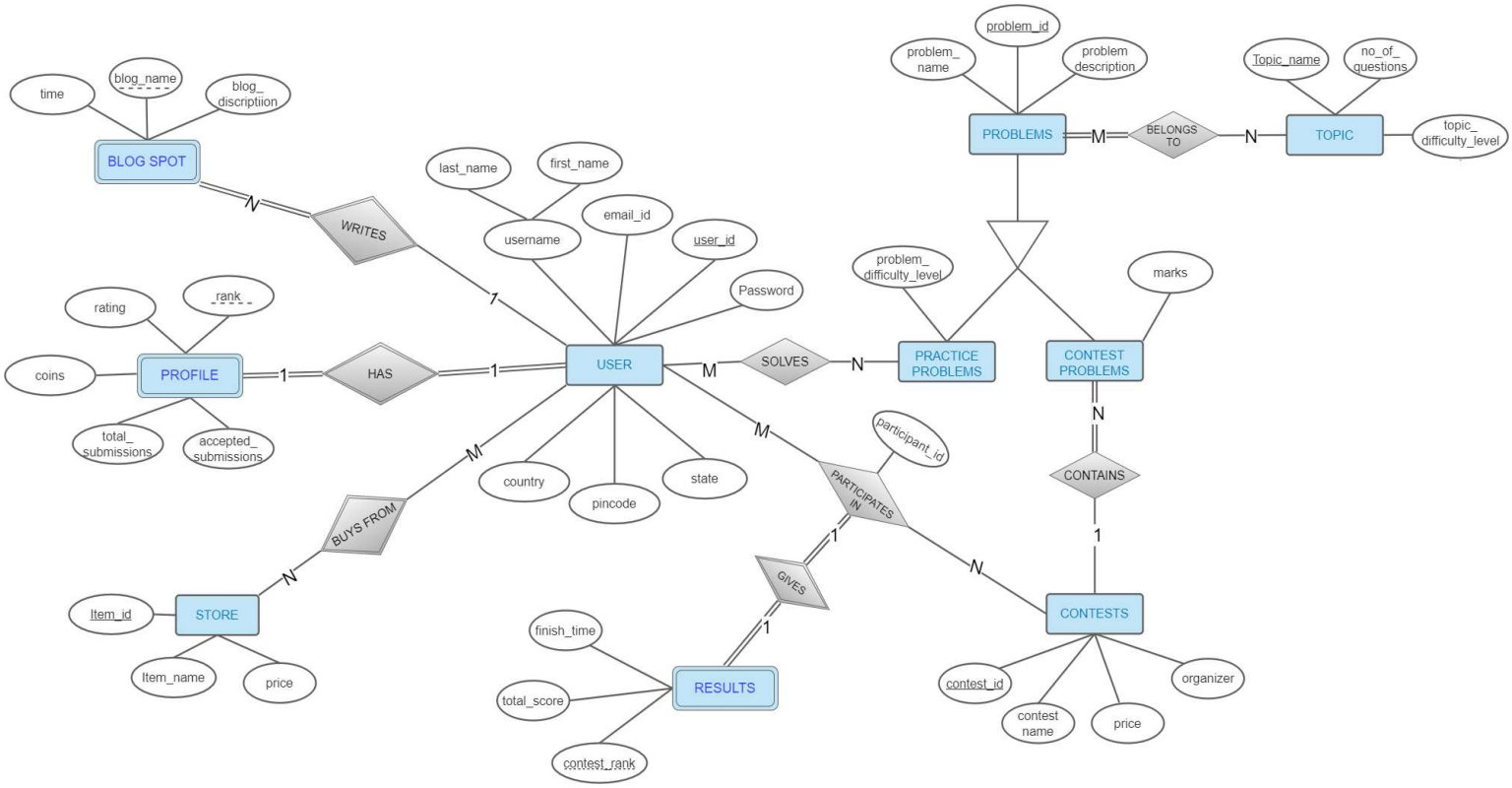
It mainly focuses on the entities and the relationships between them with all the key constraints and participation constraints.

ASSUMPTIONS :-

1. We have assumed that no two users will have the same profile.

2. A problem can include 1 or more topics.
3. A user can write any number of blogs
4. The same blog name can be used by 2 or more users.
5. No two participants will get the same result after the contest.

ER diagram :



Note :-

Let's say that a table is in 1NF if it satisfies the below condition :-

" If there are no multivalued attributes , composite attributes , all columns have unique names and here the order in which data will be stored doesn't matter. "

Lets name this condition as 1NF condition.

Relational Schema :-

Entity :- USER

Attributes :-

- User_id
- user_name(composite attribute)
- Email_id
- Password
- Country
- State
- pincode

1NF :- This relation does not satisfy 1NF condition as there is a composite attribute . inorder to bring this into 1NF we have to insert sub attributes instead of inserting main attributes.

Attributes :-

- User_id
- First_name
- last_name
- Email_id
- Password

- Country
- State
- pincode

Functional dependencies :-

- User_id ➡ first_name
- User_id ➡ last_name
- User_id ➡ email_id
- User_id ➡ password
- User_id ➡ country
- User_id ➡ state
- User_id ➡pincode
- Pincode ➡ state
- State ➡ country
- Email_id ➡ user_id
- Email_id ➡ first_name
- Email_id ➡ last_name
- Email_id ➡ country
- Email_id ➡ password
- Email_id ➡ pincode
- Email_id ➡ state

2NF :- As we can see in the above functional dependencies the determinant (X in X ➡ Y) is a single attribute i.e, all our candidate keys are single attributes . so , there exists no proper subset of candidate keys.Therefore there is no chance of existence of partial dependency . Hence this table is in 2NF.

3NF :- In the above functional dependencies we can say that pincode and state are non prime attributes and they are determining some other non prime attributes .hence we can say that there is a transitive dependency . Therefore this table is not in 3NF. To bring this table into 3NF we should do lossless decomposition.

Decomposing the above table into 3 tables.

1) **USER -** user_id, first_name, last_name, email_id, password, pincode
2) **PIN -** pincode, state
3) **STATE -** state, country

We can say that this is a valid/ lossless decomposition because the intersection of pin and state tables i.e, state determines the whole state table and their join i.e pincode, state, country when intersected with the user table gives pincode which in turn determines the join.

BCNF :- In all the above modified tables , only the superkeys are determining all other attributes . Hence, we can say that the table is in BCNF .
Note :- here it is shown as user,pin,city but the creation will be done in city,pin,user order only.

Entity :- PROFILE

Attributes :-

- User_id
- Rating
- Rank
- Coins
- Total_submissions
- accepted_submissions

Functional dependencies :-

- User_id ➡ rank
- User_id ➡ rating
- User_id ➡ coins
- User_id ➡ total_submissions
- User_id ➡ accepted_submissions

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF :- In this table there is a single candidate key i.e , user_id and it only a single attribute so there is no proper subset of this therefore there is no partial dependency. Hence this table is in 2nd normal form.

3 NF :- In this table all functional dependencies are from candidate key(prime attribute) to non prime attributes . Therefore there is no transitive dependency. Hence this table is in 3rd normal form.

BCNF :- all functional dependencies are from superkey i.e user_id to all other attributes. Therefore this table is in BCNF.

Entity :- STORE

Attributes :-

- Item_id
- Item_name
- price

Functional dependencies :-

- Item_id ➡ item_name
- Item_id ➡ price

1 NF :- this relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF :- In this table there is a single candidate key i.e , user_id and it only a single attribute so there is no proper subset of this therefore there is no partial dependency. Hence this table is in 2nd normal form.

3 NF :- In this table all functional dependencies are from candidate key(prime attribute(item_id)) to non prime attributes(item_name,price) . Therefore there is no transitive dependency. Hence this table is in 3rd normal form.

BCNF :- all functional dependencies are from superkey i.e item_id to all other attributes. Therefore this table is in BCNF.

Entity :- BUYS_FROM

Attributes :-

- user_id,item_id

There are no functional dependencies.

As there are no dependencies and all the attributes together form a primary key. So the table satisfies all normal forms conditions. Therefore its in 1,2,3,BC normal forms.

Entity :- BLOG_SPOT

Attributes :-

- User_id,blog_name
- Blog_desc
- blog_time

Functional dependencies :-

- User_id,blog_name ➡ blog_desc
- User_id,blog_name ➡ blog_time

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF :- In this table candidate key contains 2 attributes i.e, user_id and blog_name , these 2 attributes together determine blog_desc and blog_time. And part of this candidate key cannot determine others. Therefore there is no partial dependency. Hence this table is in 2nd normal form.

3 NF :- In this table all functional dependencies are from candidate key(prime attributes) i.e user_id and blog_name to non prime attributes . Therefore there is no transitive dependency. Hence this table is in 3rd normal form.

BCNF :- all functional dependencies are from superkey i.e user_id & blog_name to all other attributes. Therefore this table is in BCNF.

Entity :- PROBLEMS

Attributes :-

- Problem_id
- Problem_name
- problem_desc

Functional dependencies :-

- Problem_id ➡ problem_name
- Problem_id ➡ problem_desc

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF & 3 NF & BCNF :- In this table there are only 2 dependencies where a candidate key is determining other attributes. Therefore as the superkey is acting as determinant in all the dependencies , this table is in BCNF which means this is also in 2nd and 3rd normal forms.

Entity :- TOPIC

Attributes :-

- Topic_name
- No_of_questions
- topic_difficulty_level

Functional dependencies :-

- Topic_name ➡ no_of_questions
- Topic_name ➡ topic_difficulty_level

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF & 3 NF & BCNF :- In this table there are only 2 dependencies where a candidate key is determining other attributes. Therefore as the superkey is acting as determinant in all the dependencies , this table is in BCNF which means this is also in 2nd and 3rd normal forms.

Entity :- BELONGS_TO

Attributes :-

- problem_id,topic_name

There are no functional dependencies.

As there are no dependencies and all the attributes together form a primary key. So the table satisfies all

normal forms conditions. Therefore its in 1,2,3,BC normal forms.

Entity :- PRACTICE_PROBLEMS

Attributes :-

- Problem_id
- problem_difficulty_level

Functional dependencies :-

- Problem_id ➡ problem_difficulty_level

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF & 3 NF & BCNF :- In this table there is only one dependency where a full candidate key is determining another attribute . therefore there is no chance of existence of partial dependency . hence this table satisfies all normal forms.

Entity :- CONTEST

Attributes :-

- Contest_id

- Contest_name
- Prize
- organizer

Functional dependencies :-

- Contest_id ➡ contest_name
- Contest_id ➡ prize
- Contest_id ➡ organizer

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF & 3 NF & BCNF :- In this table there are only dependencies from primary key to all other keys which says that this table is in BCNF . Therefore this table is in 2nd 3rd normal form too.

Entity :- CONTEST_PORBLEMS

Attributes :-

- Contest_id,problem_id
- marks

Functional dependencies :-

- Contest_id,problem_id ➡ marks

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF & 3 NF & BCNF :- in this table there is only one functional dependency in which the full candidate key is determining a non prime attribute. Which satisfies the BCNF condition. Therefore the table is in 1NF,2NF,BCNF.

Entity :- PARTICIPATES_IN

Attributes :-

- User_id
- Contest_id
- participant_id

Functional dependencies :-

- User_id,contest_id ➡ participant_id
- Participant_id ➡ User_id,contest_id

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF & 3 NF & BCNF :- there are two candidate keys in this table . one is participant_id and other is user_id+contest_id as all dependencies are from one full candidate key to another so there is no existence of

partial dependency or transitive dependency . Therefore the table is in 2NF,3NF and BCNF.

Entity :- RESULTS

Attributes :-

- Participant_id
- Contest_rank
- Finish_time
- total_score

Functional dependencies :-

- Participant_id ➡ contest_rank
- Participant_id ➡ finish_time
- Participant_id ➡ total_score

1 NF :- This relation satisfies 1NF condition. Therefore it is in 1st normal form.

2 NF & 3 NF & BCNF :- In this table there are only dependencies from primary key to all other keys which says that this table is in BCNF . Therefore this table is in 2nd 3rd normal form too.

CREATING TABLES :-

1) STATE

```
Create table STATE
(
    State varchar(20),
    Country varchar(20),
    primary key(state)
 );
```

2) PIN

```
Create table PIN
(
    Pincode number(5),
    State varchar(20),
    primary key(pincode)
);
```

3) USER

```
Create table user
(
    Userid varchar(10) not null Primary key,
    Firstname varchar(20),
    Lastname varchar(20),
    Email_id varchar(30),
```

```
    Password varchar(20),
    Pincode number(5)
);
```

4) PROFILE

```
Create table Profile
(
    User_id varchar(10),
    Rating decimal(2,1),
    Rank number(4),
    Coins number,
    Total_submissions number,
    Accepted_submissions number,
    Primary key(user_id),
    Foreign key(user_id) references user(user_id)
);
```

5) STORE

```
Create table STORE
(
    Item_id number not null Primary key,
    Item_name varchar(20),
    Coins number,
);
```

6) BUYS_FROM

```
Create table buys_from
(
    Item_id number,
    User_id varchar(10),
    Primary key(user_id,item_id),
    Foreign key(user_id) references user(user_id),
    Foreign key(item_id) references store(item_id)
);
```

7) BLOG_SPOT

```
Create table blog_spot
(
    User_id varchar(10),
    Blog_name varchar(50),
    Blog_time datetime,
    Blog_desc varchar(max),
    Primary key(user_id,blog_name),
    Foreign key(user_id) references user(user_id)
);
```

8) PROBLEMS

```
Create table Problems
(
```

```
    Problem_id varchar (10) not null Primary Key ,
    Problem_name varchar(50),
    Problem_description varchar(max)
);
```

9) TOPIC

```
Create table Topic
(
    Topic_name varchar(30) not null Primary key,
    No_of_questions number,
    Diffuclty_level varchar(10)
);
```

10) BELONGS_TO

```
Create table belongs_to
(
    Problem_id varchar(10),
    Topic_name varchar(30),
    Primary key(topic_name,problem_id),
    Foreign key (Problem_id) references Problems(Problem_id),
    Foreign key (topic_name) references topic(topic_name)
);
```

11) PRACTICE_PROBLEMS

```
Create table Practice_problems
```

```
(
    Problem_id varchar(10)   not null primary key,
    Problem_Difficulty_level varchar(10) ,
    Foreign key (Problem_id) references Problems(Problem_id)
);
```

12) CONTEST

```
Create table Contest
(
    Contest_id number(4) not null primary key,
    Contest_name varchar(20),
    Prize number ,
    Organizer varchar(max)
);
```

13) CONTEST_PROBLEMS

```
Create table contest_problems
(
    Problem_id varchar(10)
    Contest_id number(4),
    Marks number,
    Primary key(Problem_id,Contest_id),
    Foreign key (Problem_id) references Problems(Problem_id),
    Foreign key (contest_id) references contest(contest_id)
```

```
);
```

14) PARTICIPATES_IN

```
Create table participates_in
(
    Contest_id number(4),
    User_id varchar(10),
    Participant_id number not null primary key,
    Foreign key (contest_id) references contest(contest_id),
    Foreign key(user_id) references user(user_id)
);
```

15) RESULTS

```
Create table Results
(
    Participant_id number  not null primary key,
    Contest_rank number,
    Total_score number,
    Finish_time datetime
);
```

Inserting data into tables :-

STATE :-

```
Insert into STATE
```

```
Values ('Telangana', 'India');
Insert into STATE
Values ('Beijing', 'China');
Insert into STATE
Values ('Tokyo', 'Japan');
Insert into STATE
Values ('Tripura', 'India');
Insert into STATE
Values ('Hassen', 'Germany');
Insert into STATE
Values ('Washington', 'America');
Insert into STATE
Values ('Islamabad', 'Pakistan');
Insert into STATE
Values ('Assam', 'India');
Insert into STATE
Values ('Shanghai', 'China');
Insert into STATE
Values ('Geneva', 'Switzerland');
```

PIN :-

```
Insert into PIN
Values (29384, 'Telangana');
Insert into PIN
```

Values (93749, ‘Beijing’);

Insert into PIN

Values (10283, ‘Tokyo’);

Insert into PIN

Values (92748, ‘Tripura’);

Insert into PIN

Values (92472, ‘Hassen’);

Insert into PIN

Values (65463, ‘Washington’);

Insert into PIN

Values (98976, ‘Islamabad’);

Insert into PIN

Values (12937, ‘Assam’);

Insert into PIN

Values (87641, ‘Shanghai’);

Insert into PIN

Values (91236, ‘Geneva’);

USER :-

Insert into user

values(’m3g4n’,’meghana’,’pasikanti’,’meghanapasikanti@gmail.com’,’12345’,29384);

Insert into user

values(’happy_1’,’bharath’,’reddy’,’bharathreddy@gmail.c

om’,’123456789’, ‘10283’);
Insert into user
values(’asam_123’,’’asam,’samba’,’asamsamba@gmail.com’,’test1’, ‘93749’);
Insert into user
values(’suji_rider’,’sujit’,’das’,’sujitdas@gmail.com’,’password’,’92748’);
Insert into user
values(’hema_123’,’hema’,’landa’,’hemalanda@gmail.com’,’12345678’,’92472’);
Insert into user
values(’sruth_56’,’sruthi’,’pulusu’,’sruthipulusu@gmail.com’,’zinch’,’65463’);
Insert into user
values(’cyberpunk’,’ashok’,’valasa’,’ashokvalasa@gmail.com’,’G_czechout’ ,’98976’);
Insert into user
values(’noob’,’rama’,’rao’,’ramarao@gmail.com’,’asdf’,’12937’);
Insert into user
values(’apsara’,’dinesh’,’reddy’,’dineshreddy@gmail.com’,’querty’,’87641’);
Insert into user

values('potassium','manoj','reddy','manojreddy@gmail.com','1234567890',91236);

Insert into user

values('icecuber','kalyan','induri','kalyaninduri@gmail.com','asdfghjkl',92748);

PROFILE :-

Insert into PROFILE

Values('m3g4n', 5 , 1, 2000, 15, 13);

Insert into PROFILE

Values('happy_1', 5 , 2, 2228 , 33, 21);

Insert into PROFILE

Values('asam_123', 4.8 , 3, 2210 , 15 , 10);

Insert into PROFILE

Values('suji_rider', 4.7 , 4, 2900 , 16, 12);

Insert into PROFILE

Values('hema_123', 4.6, 5 , 3000 , 14, 8);

Insert into PROFILE

Values('sruth_56', 4.5 , 6, 100, 12, 2);

Insert into PROFILE

Values('cyberpunk',4.4 ,7 , 200,50 ,4);

Insert into PROFILE

Values('noob',4.3 ,8 ,50 ,10 ,7);

Insert into PROFILE

Values('apsara', 4.2,9 ,300 ,10 ,7);

Insert into PROFILE

Values('potassium', 4.1,10,500 , 79, 7);

Insert into PROFILE

Values('icecuber', 4.0, 11, 270, 16,5);

STORE :-

Insert into STORE

Values(1, 'bag',1000);

Insert into STORE

Values(2, 'bottle', 800);

Insert into STORE

Values(3, 't-shirt', 500);

Insert into STORE

Values(4, 'bag', 700);

Insert into STORE

Values(5, 't-shirt', 600);

BUYS_FROM :-

Insert into buys_from

Values (2, 'm3g4n');

Insert into buys_from

```
Values (5, 'happy_1');
Insert into buys_from
Values (4, 'cyberpunk');
Insert into buys_from
Values (3, 'pottasium');
Insert into buys_from
Values (1, 'cyberpunk');
Insert into buys_from
Values (4, 'noob');
Insert into buys_from
Values (3, 'asam_123');
```

BLOG_SPOT :-

```
Insert into Blog_spot
Values ('m3g4n', 'my journey to 5*', '2020-06-23 12:32:09',
 'Hello everyone,
This is my first blog , hope you'll enjoy it .
Inspect your rating element to 5 you will become 5* coder.
Thanks!');
Insert into Blog_spot
Values (happy_1, 'my journey to 5*', '2020-07-24 12:35:09',
 'Hello everyone,
```

This is my first blog , hope you'll will enjoy it
Thanks!');
Insert into Blog_spot
Values ('cyberpunk', 'tips for coders', '2019-07-14 1:35:52',
'Hello everyone,
This is my first blog , hope you'll enjoy it .
Use Ms word for coding there you can code fastly and efficiently.
Thanks!');
Insert into Blog_spot
Values ('happy_1', 'job interview', '2018-09-04 5:35:52',
'Hello everyone,
This is my first blog , hope you'll enjoy it . nothing to say because i didn't even give interview
Thanks!');
Insert into Blog_spot
Values ('m3g4n', 'noob to pro ', '2021-11-17 23:45:45',
'Hello everyone,
This is my second blog , hope you'll enjoy it .
Start calling yourself a pro.Then you will become pro
Thanks!');

PROBLEMS :-

Note :- here we have written problem desc with very less Inorder to make it look short and clean.

Insert into problems

Values('CODA01',

'Welcome world.','Print "welcome world"')

Insert into problems

Values('CODA02', 'Beautiful String',

'Beautiful string is a string in which vowel count is equal to Consonant count.')

Insert into problems

Values('CODA03', 'Minimum no of operations','You have n boxes. You are given a binary string boxes of length n, where boxes[i] is '0' if the ith box is empty, and '1' if it contains one ball.
In one operation, you can move one ball from a box to an adjacent box. Box i is adjacent to box j if abs(i - j) == 1. Note that after doing so, there may be more than one ball in some boxes.

Return an array answer of size n, where answer[i] is the minimum number of operations needed to move all the balls to the ith box. Each answer[i] is calculated considering the initial state of the boxes.')

Insert into problems

Values('ACMU01', 'Am i prime?','check if the given number is prime or not Input :- a number t saying number of testcases . next t lines contains one number n Output :- print true if the given number is prime otherwise print false. Input :- 5 24 1123 827 623 12 Output :- no Yes Yes No no')

Insert into problems

Values('ACMU02', 'Nth tribonacci number','The Tribonacci sequence Tn is defined as follows: T0 = 0, T1 = 1, T2 = 1, and Tn+3 = Tn + Tn+1 + Tn+2 for n >= 0.Given n, return the value of Tn.

Example 1: Input: n = 4 Output: 4 Explanation: T_3 = 0 + 1 + 1 = 2 T_4 = 1 + 1 + 2 = 4')

Insert into problems

Values('ACMU03', 'Kth largest number','Given an integer array nums and an integer k, return *the* kth *largest element in the array*.Note that it is the kth largest element in the sorted order, not the kth distinct element. Input: nums = [3,2,1,5,6,4], k = 2 Output: 5 Input: nums = [3,2,3,1,2,4,5,5,6], k = 4 Output: 4')

Insert into problems

Values('ACMU04', 'Valid string','Given a binary string SS consisting of 0's0's and 1's1's, find whether there exists a rightwise circular rotation of the string such that every 2 adjacent 1 ' S 1's are separated by at most C C 0 , S 0's. Sample Input: 3 4 1 1100 4 0 0101 6 1 101001 Sample Output: YES NO YES')

Insert into problems

Values('CDR01', 'XOR array','Find the XOR of an array. Input: A vector array of size n Output: Number, i.e XOR of an array.')

Insert into problems

Values('CDR02', 'Deepest nodes sum','Given the root of a binary tree, return *the sum of values of its deepest leaves*.Input: root = [1,2,3,4,5,null,6,7,null,null,null,null,8] Output: 15')

Insert into problems

Values('CDR03', 'Is graph bipartite?','Given a graph, Find whether the graph is Bipartite.Input: A 2d array Output: Bool value (true or false).')

Insert into problems

Values('CODW01', 'Palindrome check','A number is called *palindromic* if its decimal representation is a palindrome. You are given a range, described by a pair of integers L and R. Find the sum of all palindromic numbers lying in the range [L, R], inclusive of both the extrema.Input:2 1 10 123 150 Output: 45 272')

Insert into problems

Values('CODW02', 'Sliding Window','Given a string s and an integer k. Return *the maximum number of vowel letters* in any substring of s with length k. Vowel letters in English are (a, e, i, o, u).Input: s = "abciiidef", k = 3 Output: 3 Explanation: The substring "iii" contains 3 vowel letters.')

Insert into problems

Values(DZER01, 'Reverse string','Given a sentence. Print the words of a sentence in reverse order. Sample Input: cats and dogs Sample Output: dogs and cats ')

Insert into problems

Values('DZER02', 'Spiral Order','You are given a 2d array having m rows and n columns. Write a program to print it in Spiral_form.
Input: 2 4 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 3 6 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11 ')

Insert into problems

Values('DZER03', 'Tower of hanoi','Implement Towers of Hanoi program for 3 towers(recursion), and count number of calls made.Sample Input:13 Sample Output:7')

Insert into problems

Values('KCR01', 'Water Bottles','Given numBottles full water bottles, you can exchange numExchange empty water bottles for one full water bottle.The operation of drinking a full water bottle turns it into an empty Bottle. Return the maximum number of water bottles you can drink.Input: numBottles = 9, numExchange = 3 Output: 13 ')

Insert into problems

Values('KCR02', 'LCM problem','Two integers A and B are the inputs. Write a program to find GCD and LCM of A and B. Input The first line contains an integer T, total number of testcases. Then follow T lines, each line contains an integer A and B. Output Display the GCD and LCM of A and B separated by space respectively. The answer for each test case must be displayed in a new line Input 3 120 140 10213 312 10 30')

Insert into problems

Values('KCR03', 'Smallest path','Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.Note: You can only move either down or right at any point in time. Input: grid = [[1,3,1],[1,5,1],[4,2,1]] Output: 7 Explanation: Because the path 1 → 3→ 1 → 1 → 1 minimizes the sum.Example 2: Input: grid = [[1,2,3],[4,5,6]] Output: 12')

Insert into problems

Values(‘CODE01’, ‘Binary search’,’Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search target in nums. If target exists, then return its index. Otherwise, return -1.Input: nums = [-1,0,3,5,9,12], target = 9 Output: 4’)

Insert into problems

Values(‘CODE02’, ‘Next Permutation’,’Find the next Permutation of given string. Input: A string Output: The next permutation of string’)

Insert into problems

Values(‘CODE03’, ‘Range Queries.’,’Some problems appear hard though they are very easy. Today Aakash is stuck in a range query problem. He has been given an array with only numbers 0 and 1. There are two types of queries -
0 L R : Check whether the number formed from the array elements L to R is even or odd and print EVEN or ODD respectively. Number formation is the binary number from the bits status in the array L to R
1 X : Flip the Xth bit in the array Indexing is 1 based Input
First line contains a number N and Q as input. Next line contains N space separated 0 or 1. Next Q lines contain description of each query
Output for only query type 0 L R whether the number in range L to R is "EVEN" or "ODD" (without quotes).’)

TOPIC :-

Insert into topic

values(‘Arrays’,3 ,’easy’);

Insert into topic

values(‘ Binary search’, 3,’easy’);

Insert into topic

values(‘Strings’,3 ,’easy’);

Insert into topic

values(‘Sorting’,3,’medium’);

Insert into topic

values(‘Stacks’,2 ,’easy’);

```
Insert into topic
values('Queue',2 ,'easy');
Insert into topic
values('Two Pointers',2 ,'medium');
Insert into topic
values('Depth First search',2 ,'medium');
Insert into topic
values('Greedy', 3,'medium');
Insert into topic
values('Recursion',3 ,'hard');
Insert into topic
values('Back tracking',3 ,'hard');
Insert into topic
values('Bit manipulation',2 ,'medium');
Insert into topic
values('Hashing',2 ,'hard');
Insert into topic
values('Binary Trees', 2,'hard');
Insert into topic
values('Graph',1 ,'hard');
Insert into topic
values('Linked Lists',1 ,'medium');
Insert into topic
```

values('Dynamic programming',1 ,'hard');

BELONGS_TO :-

Insert into belongs_to
Values ('CODA02' , 'Arrays');
Insert into belongs_to
Values ('KCR02' , 'Arrays');
Insert into belongs_to
Values ('KCR03' , 'Arrays');
Insert into belongs_to
Values ('KCR01' , ' Binary search');
Insert into belongs_to
Values ('KCR02' , ' Binary search');
Insert into belongs_to
Values ('CODA01' , ' Binary search');
Insert into belongs_to
Values ('CDR01' , ' Searching');
Insert into belongs_to
Values ('CODW01' , ' Searching');
Insert into belongs_to
Values ('CODW02' , ' Searching');
Insert into belongs_to

```
Values (‘CODA03’ , ‘Strings’);
Insert into belongs_to
Values (‘CDR02’ , ‘Strings’);
Insert into belongs_to
Values (‘CDR03’ , ‘Strings’);
Insert into belongs_to
Values (‘ACMU04’ , ‘Sorting’);
Insert into belongs_to
Values (‘ACMU02’ , ‘Sorting’);
Insert into belongs_to
Values (‘CODA02’ , ‘Stacks,Queue’);
Insert into belongs_to
Values (‘ACMU03’ , ‘Stacks,Queue’);
Insert into belongs_to
Values (‘ACMU02’ , ‘Two Pointers’);
Insert into belongs_to
Values (‘ACMU01’ , ‘Two Pointers’);
Insert into belongs_to
Values (‘DZER02’ , ‘Depth First search’);
Insert into belongs_to
Values (‘KCR03’ , ‘Depth First search’);
Insert into belongs_to
Values (‘DZER02’ , ‘Greedy’);
```

```
Insert into belongs_to
Values ('KCR01' , 'Greedy');
Insert into belongs_to
Values ('DZER03' , 'Greedy');
Insert into belongs_to
Values ('DZER03' , 'Recursion');
Insert into belongs_to
Values ('DZER01' , 'Recursion');
Insert into belongs_to
Values ('CODW02' , 'Recursion');
Insert into belongs_to
Values ('CODA02' , 'Back tracking');
Insert into belongs_to
Values ('CODE03' , 'Back tracking');
Insert into belongs_to
Values ('CODA03' , 'Back tracking');
Insert into belongs_to
Values ('ACMU02' , 'Bit manipulation');
Insert into belongs_to
Values ('CDR03' , 'Bit manipulation');
Insert into belongs_to
Values ('CODA02' , 'Hashing');
Insert into belongs_to
```

```
Values ('ACMU03' , 'Hashing');
Insert into belongs_to
Values ('ACMU04' , 'Binary Trees');
Insert into belongs_to
Values ('DZER03' , 'Binary Trees');
Insert into belongs_to
Values ('KCR02' , 'Graphs');
Insert into belongs_to
Values ('CODE01' , 'Linked Lists');
Insert into belongs_to
Values ('CODE03' , 'Dynamic programming');
```

PRACTICE_PROBLEMS :-

```
Insert into practice_problems
values('CODA01','easy');
Insert into practice_problems
values('CODA02','medium');
Insert into practice_problems
values('CODA03','hard');
Insert into practice_problems
values('ACMU01','easy');
Insert into practice_problems
```

```
values('ACMU02','easy');
Insert into practice_problems
values('ACMU03','medium');
Insert into practice_problems
values('ACMU04','medium');
Insert into practice_problems
values('CDR01','easy');
Insert into practice_problems
values('CDR02','easy');
Insert into practice_problems
values('CDR03','hard');
Insert into practice_problems
values('CODW01','medium');
Insert into practice_problems
values('CODW02','hard');
Insert into practice_problems
values('DZER01','easy');
Insert into practice_problems
values('DZER02','medium');
Insert into practice_problems
values('DZER03','hard');
Insert into practice_problems
values('KCR01','easy');
```

```
Insert into practice_problems
values('KCR02','hard');
Insert into practice_problems
values('KCR03','medium');
Insert into practice_problems
values('CODE01','medium');
Insert into practice_problems
values('CODE02','hard');
Insert into practice_problems
values('CODE03','easy');
```

CONTEST :-

```
Insert into contest
values(0001,'Codeathon 3.o',1500 ,'CSEA');
Insert into contest
values(0002,'ACM Unlock', 1200,'ECEA');
Insert into contest
values(0003,'Coderbyte',2000 ,'IIIH');
Insert into contest
values(0004,'Code wars 4.0',1000 ,'Harbour Space Univer
sity');
```

```
Insert into contest
values(0005,'Divide By zero',10000 ,'Harvard University');
Insert into contest
values(0006,'Kick Start',5000 ,'CodeNation');
Insert into contest
values(0007,'Hash Code',2500 ,'CSEA');
```

CONTEST_PROBLEMS :-

```
Insert into contest_problems
Values ('CODA01',0001 , 100);
Insert into contest_problems
Values ('CODA02', 0001, 200);
Insert into contest_problems
Values ('CODA03', 0001, 300);
Insert into contest_problems
Values ('ACMU01',0002 ,100 );
Insert into contest_problems
Values ('ACMU02',0002 ,150 );
Insert into contest_problems
Values ('ACMU03', 0002,150 );
Insert into contest_problems
Values ('ACMU04', 0002,200 );
Insert into contest_problems
Values (CDR01, 0003, 150);
```

```
Insert into contest_problems
Values ('CDR02',0003 , 150);
Insert into contest_problems
Values ('CDR03', 0003, 200);
Insert into contest_problems
Values ('CODW01', 0004,200);
Insert into contest_problems
Values ('CODW02',0004 , 300);
Insert into contest_problems
Values ('DZER01',0005 ,400);
Insert into contest_problems
Values ('DZER02',0005 ,300 );
Insert into contest_problems
Values ('DZER03',0005 ,400 );
Insert into contest_problems
Values ('KCR01',0006 ,200);
Insert into contest_problems
Values ('KCR02', 0006, 150);
Insert into contest_problems
Values ('KCR03',0006 , 350);
Insert into contest_problems
Values ('CODE01',0007 ,400 );
Insert into contest_problems
```

Values (‘CODE02’,0007 ,150);
Insert into contest_problems
Values (‘CODE03’,0007 , 300);

PARTICIPATES_IN :-

Insert into participates_in
Values (0001 ,’m3g4n’,2021001);
Insert into participates_in
Values (0001,’happy_1’,2021002);
Insert into participates_in
Values (0001 ,’asam_123’,2021003);
Insert into participates_in
Values (0002 ,’happy_1’,2021004);
Insert into participates_in
Values (0002 ,’m3g4n’,2021005);
Insert into participates_in
Values (003,’asam_123’,2021006);
Insert into participates_in
Values (0003,’suji_rider’,2021007);
Insert into participates_in
Values (0003 ,’hema_123’,2021008);
Insert into participates_in
Values (0004,’hema@123’,2021009);

```
Insert into participates_in
Values ( 0004,'cyberpunk',2021010);
Insert into participates_in
Values ( 0004,'icecuber',2021011);
Insert into participates_in
Values ( 0005,'m3g4n',2021012);
Insert into participates_in
Values ( 0005,'happy_1',2021013);
Insert into participates_in
Values ( 0005,'noob',2021014);
Insert into participates_in
Values ( 0005,'suji_rider',2021015);
Insert into participates_in
Values (0005 ,'icecuber',2021016);
Insert into participates_in
Values ( 0005,'potassium',2021017);
Insert into participates_in
Values ( 0006,'hema_123',2021018);
Insert into participates_in
Values ( 0006,'sruth_56',2021019);
Insert into participates_in
Values (0006 ,'m3g4n',2021020);
Insert into participates_in
```

```
Values (0007 ,'apsara',2021021);
Insert into participates_in
Values ( 0007,'asam_123',2021022);
Insert into participates_in
Values ( 0007,'noob',2021023);
Insert into participates_in
Values ( 0007,'cyberpunk',2021024);
```

RESULTS :-

```
Insert into Results
Values ( 2021001, 1,600 , '2020-08-23 12:45:03');
Insert into Results
Values ( 2021002, 2,400 , '2020-08-23 21:50:43');
Insert into Results
Values ( 2021003, 3,350 , '2020-08-23 08:00:03');
Insert into Results
Values ( 2021004,1 , 400, '2019-08-22 05:46:03');
Insert into Results
Values (2021005 ,2 ,250 , '2019-08-20 05:46:03');
Insert into Results
Values (2021006 , 3,450 , '2019-10-22 05:00:45');
```

```
Insert into Results
Values (2021007 ,2 ,350 , '2019-10-22 12:13:33');
Insert into Results
Values (2021008 ,1,450 , '2019-10-22 05:25:41');
Insert into Results
Values (2021009 , 1, 600, '2018-04-15 04:41:45');
Insert into Results
Values (2021010 , 2,750 , '2018-04-16 05:41:45');
Insert into Results
Values ( 2021011, 3,400 , '2018-04-17 04:25:45');
Insert into Results
Values ( 2021012,1 , 700, '2021-04-15 15:42:45');
Insert into Results
Values ( 2021013, 2,500 , '2021-04-19 19:19:19');
Insert into Results
Values (2021014 , 3,550 , '2021-04-13 13:31:45');
Insert into Results
Values (2021015 ,4 ,600 , '2021-04-15 14:41:00');
Insert into Results
Values (2021016 ,5 ,650 , '2021-04-17 23:54:45');
Insert into Results
Values ( 2021017,6 ,400 , '2021-04-15 04:32:15');
Insert into Results
```

```
Values ( 2021018,1 ,350, '2017-02-15 21:32:15');
Insert into Results
Values ( 2021019,2 ,450, '2017-02-15 04:39:15');
Insert into Results
Values (2021020 ,3 ,550 , '2017-02-15 13:52:15');
Insert into Results
Values ( 2021021,1 ,500 , '2019-07-15 14:05:00');
Insert into Results
Values ( 2021022,2 , 450, '2019-07-14 02:05:00');
Insert into Results
Values ( 2021023,3 ,450 , '2019-07-16 09:09:03');
Insert into Results
Values ( 2021024,4 ,300 , '2019-07-14 00:00:07');
```

Few tables output :-

USER :-

	USERID	FIRSTNAME	LASTNAME	EMAIL_ID	PASSWORD1	PINCODE
1	m3g4n	meghana	pasikanti	meghanapasikanti@gmail.com	12345	29384
2	happy_1	bharath	reddy	bharathreddy@gmail.com	123456789	10283
3	asam_123	asam	samba	asamsamba@gmail.com	test1	93749
4	suji_rider	sujit	das	sujitdas@gmail.com	password	92748
5	hema_123	hema	landa	hemalanda@gmail.com	12345678	92472
6	sruth_56	sruthi	pulusu	sruthipulusu@gmail.com	zinch	65463
7	cyberpunk	ashok	valasa	ashokvalasa@gmail.com	G_czechout	98976
8	noob	rama	rao	ramarao@gmail.com	asdf	12937
9	apsara	dinesh	reddy	dineshreddy@gmail.com	querty	87641
10	potassium	manoj	reddy	manojreddy@gmail.com	1234567890	91236
11	icecuber	kalyan	induri	kalyaninduri@gmail.com	asdfghjkl	92748

PIN :-

	STATE1	COUNTRY
1	Telangana	India
2	Beijing	China
3	Tokyo	Japan
4	Tripura	India
5	Hassen	Germany
6	Washington	America
7	Islamabad	Pakistan
8	Assam	India
9	Shanghai	China
10	Geneva	Switzerland

STATE :-

	PINCODE	STATE
1	29384	Telangana
2	93749	Beijing
3	10283	Tokyo
4	92748	Tripura
5	92472	Hassen
6	65463	Washington
7	98976	Islamabad
8	12937	Assam
9	87641	Shanghai
10	91236	Geneva

PROFILE :-

	USER_ID	RATING	RANK1	COINS	TOTAL_SUBMISSIONS	ACCEPTED_SUBMISSIONS
1	m3g4n	5	1	2000	15	13
2	happy_1	5	2	2228	33	21
3	asam_123	4.8	3	2210	15	10
4	suji_rider	4.7	4	2900	16	12
5	hema_123	4.6	5	3000	14	8
6	sruth_56	4.5	6	100	12	2
7	cyberpunk	4.4	7	200	50	4
8	noob	4.3	8	50	10	7
9	apsara	4.2	9	300	10	7
10	potassium	4.1	10	500	79	7
11	icecuber	4	11	270	16	5