**Java Code:**

**1) Java Arraylist**

Input Format

The first line has an integer . In each of the next  lines there will be an integer  denoting number of integers on that line and then there will be  space-separated integers. In the next line there will be an integer  denoting number of queries. Each query will consist of two integers  and .

Constraints

1<=n<=20000

0<=d<=50000

1<=q<=1000

1<=x<=n

Each number will fit in signed integer.

Total number of integers in  lines will not cross 100000.

Output Format

In each line, output the number located in  position of  line. If there is no such position, just print "ERROR!"

Sample Input

5

5 41 77 74 22 44

1 12

4 37 34 36 52

0

3 20 22 33

5

1 3

3 4

3 1

4 3

5 5

Sample Output

74

52

37

ERROR!

ERROR!

**Solution:**

**package** Assessment;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** ArrayList {

**public** **static** **void** main(String[] args) {

Scanner s = **new** Scanner(System.***in***);

**int** n = s.nextInt();

ArrayList<ArrayList<Integer> > arr =

**new** ArrayList<ArrayList<Integer> >(n);

**for**(**int** i=0;i<n;i++) {

**int** num = s.nextInt();

ArrayList<Integer> a = **new** ArrayList<Integer>();

**for**(**int** k=0;k<num;k++) {

a.add(s.nextInt());

}

arr.add(a);

}

System.***out***.println("The output is");

**int** q = s.nextInt();

**for**(**int** i=0;i<q;i++) {

**int** x = s.nextInt()-1;

**int** y = s.nextInt()-1;

**if**(y>(arr.get(x).size()) - 1) {

System.***out***.println("ERROR!");

}

**else** {

System.***out***.println(arr.get(x).get(y));

}

}

}

}

**2) Java List**

Input Format

The first line contains an integer,  (the initial number of elements in ).

The second line contains  space-separated integers describing .

The third line contains an integer,  (the number of queries).

The  subsequent lines describe the queries, and each query is described over two lines:

If the first line of a query contains the String Insert, then the second line contains two space separated integers , and the value  must be inserted into  at index .

If the first line of a query contains the String Delete, then the second line contains index , whose element must be deleted from .

Constraints

1<=N<=4000

1<=Q<=4000

Each element in is a 32-bit integer.

Output Format

Print the updated list  as a single line of space-separated integers.

Sample Input

5

12 0 1 78 12

2

Insert

5 23

Delete

0

Sample Output

0 1 78 12 23

**SOLUTION:**

**3) Java Map**

You are given a phone book that consists of people's names and their phone number. After that you will be given some person's name as query. For each query, print the phone number of that person.

Input Format

The first line will have an integer  denoting the number of entries in the phone book. Each entry consists of two lines: a name and the corresponding phone number.

After these, there will be some queries. Each query will contain a person's name. Read the queries until end-of-file.

Constraints:

A person's name consists of only lower-case English letters and it may be in the format 'first-name last-name' or in the format 'first-name'. Each phone number has exactly 8 digits without any leading zeros.

1<=n<=100000

1<=Query<=100000

Output Format

For each case, print "Not found" if the person has no entry in the phone book. Otherwise, print the person's name and phone number. See sample output for the exact format.

To make the problem easier, we provided a portion of the code in the editor. You can either complete that code or write completely on your own.

Sample Input

3

uncle sam

99912222

tom

11122222

harry

12299933

uncle sam

uncle tom

harry

Sample Output

uncle sam=99912222

Not found

harry=12299933

**SOLUTION:**

**import** java.util.HashMap;

**import** java.util.Scanner;

**public** **class** MapImpl {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

HashMap<String, Integer> hash = **new** HashMap<>();

Scanner sc = **new** Scanner(System.***in***);

**int** n=sc.nextInt();

sc.nextLine();

**for**(**int** i=0;i<n;i++)

{

String name=sc.nextLine();

**int** phn=sc.nextInt();

sc.nextLine();

hash.put(name,phn);

}

**while**(sc.hasNext())

{

String str=sc.nextLine();

**try**

{

**int** out=hash.get(str);

System.***out***.println(str+"="+out);

}

**catch**(Exception e)

{

System.***out***.println("Not found");

}

}

}

}

**Coding MCQ practice set:**

Graphical user interface, text, application

Description automatically generated

Answer:

Aarushi

15

Table

Description automatically generated with medium confidence

Answer:

Set 1: [A, B, C, D]

Set 1: [E]

Set 1: [A, B, C, D, E]

Set 1: [A, B, C]

Graphical user interface, text

Description automatically generated

Answer: Option A – Error, 2

Text

Description automatically generated

Q14 Answer : B – You cannot specify which thread will get notified

Graphical user interface, text, application

Description automatically generated

Answer:

1

2

3

1

1

2

2

3

3

Graphical user interface, text, application, email

Description automatically generated

Answer: Option C – does not compile as it cannot be referenced in static method

Graphical user interface, text, application

Description automatically generated

Ans

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Ans

**- Capgemini SQL assessment:**

Text

Description automatically generated

Text

Description automatically generated

**Written query:**

1. WAQ to display second highest salary in HR schema.

query: select MAX(salary) from employees where salary < (select MAX(salary) from employees);

2. WAQ to display name of employee who is earning maximum in his/her department.

query: SELECT department\_id, first\_name, salary FROM employees d WHERE salary = (SELECT MAX(salary) FROM employees WHERE department\_id = d.department\_id);

3. WAQ to display emplyees count who are working from same location.

query: select l.city , count(e.first\_name) as Employees from employees e, departments d , locations l where e.department\_id = d.department\_id and d.location\_id = l.location\_id group by city;

4. WAQ to display number of employees joined year wise.

query: SELECT to\_char(hire\_date, 'yyyy') as YEAR , COUNT(first\_name) as Employees FROM employees GROUP BY to\_char(hire\_date, 'yyyy') ORDER BY to\_char(hire\_date, 'yyyy') ;

5. WAQ to top 2 earning employee name and salary in each department.

query: select first\_name , salary from employees order by salary desc limit 2;