## Coding Problems for practice

## 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** ttest;

**import** java.util.\*;

**public** **class** Solution {

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

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

**int** numLines = Integer.*parseInt*(sc.nextLine());

ArrayList<ArrayList> listArray = **new** ArrayList<ArrayList>();

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

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

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

**for**(**int** j=0;j<numOfIntegers;j++){

intArrayList.add(**new** ~~Integer~~(sc.nextInt()));

}

listArray.add(intArrayList);

sc.nextLine();

}

**int** numQueries = Integer.*parseInt*(sc.nextLine());

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

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

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

sc.nextLine();

**if**(x<listArray.size() && y<listArray.get(x).size()){

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

}**else**{

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

}

}

}

}

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 :

package ttest;

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

public class JavaList

{

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int numElements = Integer.parseInt(sc.nextLine());

ArrayList<Integer> numList = new ArrayList<>();

for(int i = 0;i < numElements;i++){

numList.add(sc.nextInt());

}

int numQueries = sc.nextInt();

sc.nextLine();

for(int i = 0;i < numQueries;i++){

String queryType = sc.nextLine();

if(queryType.equals("Insert")){

String[] queryArray = sc.nextLine().split(" ");

numList.add(Integer.parseInt(queryArray[0]),Integer.parseInt(queryArray[1]));

}else{

int removeIndex = Integer.parseInt(sc.nextLine());

numList.remove(removeIndex);

}

}

for(Integer num : numList){

System.out.print(num+" ");

}

}

}

## 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 :

package ttest;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class Phone {

public static void main(String[] args) {

Map<String, String> phoneBook = new HashMap<String, String>();

Scanner sc = new Scanner(System.in);

int numFriends = sc.nextInt();

sc.nextLine();

for (int i = 0; i < numFriends; i++) {

String name = sc.nextLine();

String phone = sc.nextLine();

phoneBook.put(name, phone);

}

while (sc.hasNext()) {

String inputName = sc.nextLine();

if (phoneBook.containsKey(inputName)) {

System.out.println(inputName + "=" + phoneBook.get(inputName));

} else {

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

}

}

sc.close();

}

}