Calgary Collegiate Programming Contest

University of Calgary

October 3, 2009

EVENT SPONSOR

Department of Computer Science

CONTEST ORGANIZERS

Darko Aleksic

Dr. Peter Høyer

Sean McIntyre

PROBLEM AUTHORS

Darko Aleksic

Dr. Peter Høyer

Sean McIntyre

John Zhang

RULES & INSTRUCTIONS

The following should be observed for the duration of the contest:

General Conduct

1. You may communicate with your teammates and contest officials only.
2. Use of any electronic device other than your assigned workstation is prohibited. (Devices such as watches are permitted.)
3. You may use as reference any printed materials you brought to the contest.
4. No food or drinks are allowed at the workstations!!!

Contest Computer & Resources

1. You may use only the one workstation assigned to you.
2. You may use any tools, including text editors, IDEs, debuggers, and other support applications that are installed on the contest workstation.
3. Access to the internet and network are prohibited except to the official contest page.
4. If you encounter technical difficulties, contact a contest official immediately.

Creating Solutions

1. You may program your solutions in standard C, C++, or Java.
2. Your program should read from standard in and write to standard out.
3. Your program must process the judge input data file in 10 seconds or less.
4. The entire source code for your solution must be in a single source file. If using Java, the main method must be in a class with the specified name (other inner classes are allowed).
5. When you are finished with a program, submit the source code for your solution on the contest page using your contest ID and password.
6. Questions about the problem statements should be sent via the clarifications request form on the contest page.

Enjoy the contest!

CHEAT SHEET

inputFile contents

```
1 2 3
```

Java

```
// file: "X.java"; compile: "javac X.java"; run: "java X < inputFile"
import java.io.*;
import java.util.*; // check Java API for useful Scanner methods
public class X {
	public static void main(String[] args) throws IOException {
		BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
		Scanner sc = new Scanner(in.readLine());
		while (sc.hasNext()) System.out.print(sc.next() + " ");
		System.out.println();
	}
} // output: "1 2 3 "
```

C

```
// file: "X.c"; compile: "gcc -Wall X.c"; run: "./a.out < inputFile"
#include <stdio.h>
int main() {
	int x, y, z;
	scanf("%d %d %d", &x, &y, &z);
	printf("z: %d, y: %d, x: %d\n", z, y, x);
	return 0;
} // output: "z: 3, y: 2, x: 1"
```

C++

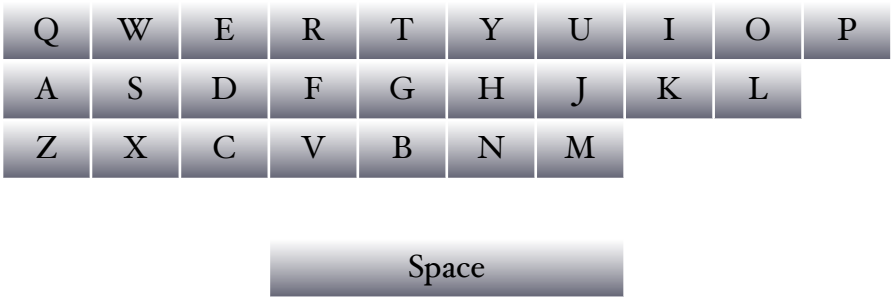
```
// file: "X.cpp"; compile: "g++ -Wall X.cpp"; run: "./a.out < inputFile"
#include <string>
#include <iostream>
#include <sstream>
using namespace std;
int main() {
	string line; int x, y, z;
	getline(cin, line);
	stringstream in(line);
	in >> x >> y >> z;
	cout << "z: " << z << ", y: " << y << ", x: " << x << endl;
	return 0;
} // output: "z: 3, y: 2, x: 1"
```

Problem A

ALBERT'S ABILITY

File name: `A.{java,c,cpp}`

You've built a keyboard for your talented pet elephant, Albert, who uses his trunk to type letters and his right foot to hit the space bar. You've laid it out in the following form.



The layout of the keyboard you've built for Albert.

It takes Albert 2 seconds to move his trunk from one key to a neighboring key (horizontally or vertically only), 1 second to press a key, and 5 seconds to press the space bar using his foot. Albert starts with his trunk over the 'E' key for each test case.

Given a string consisting only of uppercase letters A-Z and spaces, output how many seconds it takes the elephant to key in the string.

Program Input

The first line of input is the number of test cases T ($1 \le T \le 100$). The following lines will contain the test cases, one per line. No line will contain more than 100 characters.

Program Output

For each test case, output on a new line how many seconds it takes Albert to type the input.

Sample Input & Output

INPUT	OUTPUT
3 HELLO E IS FOR ELEPHANT KNOCK ON THE SKY AND LISTEN TO THE SOUND	37 161 338

Problem B

IGOR'S ANTS

File name: `B.{java,c,cpp}`

N ants are placed on a circular track of length *N* units at 0, 1, ..., *N*-1 units clockwise from an arbitrary spot. At the same time, all ants start moving in either direction at the speed of one unit per second. When two ants bump into each other, they both change direction and continue moving.

Where is each ant and what direction is it facing after *N* seconds?

Program Input

Input starts with a single integer *C* (1 ≤ *C* ≤ 100) on a line, which denotes the number of test cases to follow. Each test case consists of two lines. The first line contains a positive integer *N* (1 ≤ *N* ≤ 100). The integer *N* denotes the number of ants and it also denotes the length of the circular track. The second line contains *N* integers separated by spaces. The *i*th integer on this line determines the initial direction of the *i*th ant, 0 denoting clockwise and 1 denoting the counterclockwise direction.

Program Output

For each test case print a single line containing *N* pairs of integers separated by spaces. The *j*th pair of integers will denote the current position of the *j*th ant and the direction it is facing.

Sample Input & Output

INPUT	OUTPUT
2 2 0 1 3 0 1 0	0 0 1 1 1 1 2 0 0 0

Problem C

COMPUTER CORD

File name: `C.{java,c,cpp}`

One day Alex and Sonny were at school on their laptops practicing for the 2009 Rocky Mountain Regionals, but Alex forgot his laptop power cord. Luckily for Alex, Sonny's laptop was similar and he brought his power cord, so they took turns charging with one cord.

Alex posed the following questions: "If we charge optimally, will both our laptops run out of batteries? If not, what is longest amount of time both our laptops stay powered?"

	Initial Charge	Charging Rate	Consumption Rate
Alex	*A* units	*B* units per minute	*C* units per minute
Sonny	*D* units	*E* units per minute	*F* units per minute

A description of Alex's and Sonny's batteries.

A laptop runs out of batteries when it reaches 0 units of charge, however is still powered with an infinitesimally small value greater than 0. A battery cannot have more than 100 units of charge. For simplicity assume it takes no time to change the cord between laptops.

Program Input

The first line of the input contains an integer T ($1 \le T \le 100$), the number of test cases. Each test case is a line of integers "$A\ B\ C\ D\ E\ F$" (see table). All values will be between 1 and 100.

Program Output

If there is a way to charge so that both batteries will not run out, output "Sonny and Alex are safe". Otherwise, output the maximum time in minutes both laptops do not run out of batteries rounded to the nearest minute.

Sample Input & Output

Input	Output
1 10 10 10 20 10 20	7

Problem D

MANHATTAN HEAT

File name: `D.{java,c,cpp}`

Brett is visiting NYC, but gets caught in the middle of the humid heat. There are N notable locations in Manhattan that have air conditioning, he'd like to visit M of them. Traveling between the locations requires him to be outside in the heat. The Cartesian coordinates of the N locations are given, which M of them, and in what order, should he visit so that he is outside for the shortest distance? Remember that in Manhattan, you can only travel north and south or east to west. Also, suppose that Brett can start at and leave from any one of the N locations (he has a taxi drop him off and pick him up).

Program Input

The input format is as follows:

```
T
N M
x0 y0
x1 y1
...
xN-1 yN-1
...
```

Where T ($1 \le T \le 20$) is the number of test cases, and x_i, y_i ($1 <= x_i, y_i <= 1000$) are the x and y coordinates of each of the N ($1 \le N \le 8$) notable locations, M ($1 \le M \le N$) of which Brett would like to visit.

Program Output

The output for each test case should be on its own line, in the following format:

```
L0 L1 L2 ... LM-1
```

Where L_o is the index (0-based) of the location that Brett should visit in order from left to right. If more than one route is minimal, use the one that comes *lexicographically first*.

P	E	A	B	O	D	Y	
P	E	A	N	U	T		
P	E	A	N	U	T	T	Y
0	1	2	3	4	5	6	7

We call the above words lexicographically ordered from top to bottom because, in ASCII, B < N from column 3 and peanutty is longer than peanut. We call peabody lexicographically first. *Many library sorts such as Java's String comparator, Arrays.sort(), Collections.sort() and C++'s sort() order strings lexicographically by default.*

Sample Input & Output

Input	Output
2 4 3 8 3 6 4 18 3 12 12 4 2 10 11 15 20 16 6 9 8	1 0 2 0 3

Problem E

CRYPTIC FRAGMENTS

File name: `E.{java,c,cpp}`

- "Calgary Collegiate Programming Contest"
- "Compete in Java, C, C++"
- "Visit our web page for more details"

As exciting as these sentence fragments look on a poster, they don't exactly get people lining up to participate. Therefore the contest organizers decided to come up with one more enticing fragment to add to their poster. They decided it would be a space-separated subset of the following words.

- "Challenges"
- "Confusion"
- "Enlightenment"
- "Fragrance"
- "Frustration"
- "Glory"
- "Pizza"
- "Pop"
- "Prizes"
- "Tiredness"

A not so cryptic poster

After many hours of contemplating which space-separated subset to use, one organizer finally suggested the fragment be clever, mysterious, intriguing, and alluring. The fragment should be *cryptic-ordered*!

The acronym of "Calgary Collegiate Programming Contest", CCPC, is a *cryptic-friendly* acronym because it is composed of exactly two letters (in this case C's and P's). CCPC's *cryptic acronym* is PPCP because their two letters are exchanged. There are a number of fragments composed of space-separated subsets from the above list whose acronym is PPCP, including "Pop Prizes Confusion Pizza" and "Prizes Pizza Challenges Pop", all of which are called *cryptic* because their acronym is the *cryptic acronym*. The lexicographically-first *cryptic fragment*, "Pizza Pop Challenges Prizes" in this case, is called the *cryptic-ordered fragment*.

Therefore "Pizza Pop Challenges Prizes" was added to the poster and the results were evident. Just look around you.

Given a title and a list of words, output the title's *cryptic-ordered fragment*.

Program Input

The input begins with T ($1 \le T \le 100$) on its own line, the number of test cases. For each test case, the first line are integers N ($1 \le N \le 100$) and M ($1 \le M \le 100$), N being the number of words in the title and M being the number of words in the list. The second line contains the title, N words separated by a single space. The next M lines contain one list word each, none of which are repeated in the same test case. Each list word and title word are alphabetic: the first letter is upper case, and the rest are lowercase.

Program Output

Output the title's *cryptic-ordered fragment*. If the title does not have a *cryptic-friendly acronym*, output "No cryptic-friendly acronym" on its own line. If there is no *cryptic-ordered fragment*, "No cryptic-ordered fragment" on its own line.

Sample Input & Output

Input	Output
3 4 3 Alberta Collegiate Programming Contest Easy Edmonton Upcoming 3 3 Alligator Appreciation Club Crocodilic Animals Consume 4 3 Order Of The Termites Take Out Time	No cryptic-friendly acronym Consume Crocodilic Animals No cryptic-ordered fragment

Problem F

QUEEN OF SPADES

File name: `F.{java,c,cpp}`

You're given a pile of N playing cards. K of the N cards are the Queen of Spades. You draw two cards from the pile uniformly at random. The probability that both cards are the Queen of Spades is A/B for some integers A and B.

Given A and B, output N and K.

Program Input

The first line of input is the number of test cases T ($1 \le T \le 100$). The next T lines contain a test case of the format "A B". The values of A and B will each fit in 32-bit integers.

Program Output

For each test case, output the pair of integers N ($2 \le N \le 10^6$) and K ($2 \le K \le N$) on a line separated by a single space. If more than one solution exists, output the one with lowest N.

Sample Input & Output

INPUT	OUTPUT
3 3 4 1 6 37 101	8 7 4 2 34846 21091

Problem G

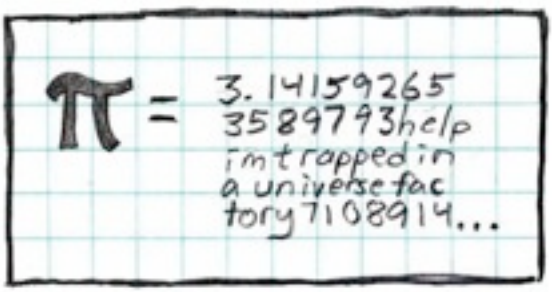
YET ANOTHER SEQUENCE

File name: `G.{java,c,cpp}`

You are given the first N elements of a sequence of decimal digits. Starting with $(N+1)$th digit, every element is the last digit of the sum of the previous N elements. What is the Kth digit in this sequence?

Program Input

Input starts with a single integer C ($1 \le C \le 100$) on a line, which denotes the number of test cases to follow.



Each test case consists of two lines . The first line starts with a positive integer N ($4 \le N \le 8$), the number from the description above. N decimal digits follow on the same line, separated by spaces. The second line starts with a positive integer M ($1 \le M \le 100$), the number of queries. M integers follow, each one corresponding to K in the statement ($0 \le K \le 10^{15}$) where K is a zero-based index, also separated by spaces.

Program Output

For each K given, output the Kth element of the sequence in that test case on its own line.

Sample Input & Output

INPUT	OUTPUT
2 4 1 2 3 4 1 4 5 1 1 1 1 1 2 8 9	0 3 5