

Departamento de Informática
Faculdade de Ciências
Universidade de Lisboa

TIUP 2009

TORNEIO INTER-UNIVERSITÁRIO DE PROGRAMAÇÃO

SECOND STAGE

Lisboa, 22-April-2009
17:00 – 20:00

<http://mooshak.di.fc.ul.pt/~tiup09>

Problems

- A – The Gates of Atlantis
- B – Mutant Knights
- C – Exam Scheduling
- D – Gesture Recognition
- E – TIUPix's Electronic Invoices

Scientific Committee

Francisco Couto <fcouto@di.fc.ul.pt>

Carlos Duarte <cad@di.fc.ul.pt>

Luís Cruz-Filipe <lcf@di.fc.ul.pt>

Hugo Miranda <hmiranda@di.fc.ul.pt>

Isabel Nunes <in@di.fc.ul.pt>

João Sarmiento <jsarmiento@gmail.com>

Paulo Sousa <pjsousa@di.fc.ul.pt>

General Information

1. The contest follows ICPC rules (to know more, go to <http://icpc.baylor.edu/>).
2. It has a duration of 3 hours for the 5 problems.
3. The programs must read from the standard-input and write to the standard-output.
4. The generated test inputs use the following norms:
 - There are no spaces at the end of the lines and each line ends with a carriage return.
 - No multiple spacing is being used, unless it is explicitly mentioned on the problem description.
5. The generated test outputs follow the same rules.
6. All problems have 1 second timeout.
7. Each team (3 persons maximum) should use one PC and it is strictly forbidden to use any online resource other than Mooshak or what is given in the local PC disk.
8. Further information can be obtained by clicking on “Help” on your Mooshak account screen.
9. Please ask your questions to the jury using the “Questions” button.

Compilers

Language	Compiler	Version	Command Line	Extension
C	gcc	4.2.4	gcc -ansi -lm \$file	.c
C++	gcc	4.2.4	g++ -Wall \$file	.cpp
Java	jdk	1.6.0_11	javac -nowarn \$file	.java
Pascal	Free Pascal	2.2.2	fpc -v0w -oprog \$file	.pas
Haskell	Hugs 98	September 2006	runhugs \$file	.hs

Problem A

The Gates of Atlantis

Introduction

Atlantis was an ancient civilisation involved in such a shade of mystery that not even its location is known. According to the legend, only those that devoted their life to knowledge were authorised to visit the city so that the inhabitants did not get polluted with the less intellectual habits of the other civilisations. To verify if some deserved to enter, the gates of Atlantis were guarded by an Oracle responsible for deciding on the faith of the tentative visitors. Those capable of answering to the questions of the Oracle were accepted. The remaining, in fact the vast majority of visitors, were condemned to death by the purifying flames of knowledge of a nearby volcano.

Recently, a group of archaeologists found, under the sands of the Sahara desert, a stone written by one of the few that, somehow, survived to the punishment. According to the writing, he was challenged with a multiplication puzzle. The puzzle presented all the steps of a multiplication of 3 by 2 digits. However, each digit was replaced by a symbol as in the example below (where symbols are replaced by characters):

$$\begin{array}{r}
 B B A \\
 \times D C \\
 \hline
 E F F A \\
 E C G E \\
 \hline
 E D G E A
 \end{array}$$

According to the writing, visitors would be allowed to enter if they were able to map each symbol on a digit so that the multiplication was correct.

Due to the apparent impossibility to solve the problem, archaeologists initially considered the stone as an hoax. However, a computer engineer which became aware of the problem proved that solving it is indeed possible. . .

Problem

Given the parcels of a multiplication where all occurrences of the same digit were replaced by the same letter, find the correspondence between each letter and a digit that makes the multiplication correct. There is a one to one mapping of digits on letters. The leftmost character of any row is guaranteed not to be zero. All puzzles multiply a number x such that ($100 \leq x < 1000$) by a number y such that ($10 \leq y < 100$). It is guaranteed that all problems presented will have exactly one solution.

Input

The first input line contains the number of multiplication puzzles N to be presented. This line is followed by $5 \times N$ lines describing the puzzles. Puzzles are presented in 5 consecutive lines, each line with the letters on one row of the puzzle, from left to right (i.e., the first letter

will correspond to the most significant digit on that row). All input is composed of upper case characters without white spaces.

The rows follow the order presented above: first the operands, then the partials and finally the result. The number of letters required on each puzzle may vary. In all problems the letters used will follow the alphabetical order starting from A.

Output

The output of each puzzle is a single line describing a mapping of letters on digits. Each mapping is represented by the letter (in upper case), followed by a white space, followed by the corresponding digit, followed by another white space. A white space should not follow the last digit. The mapping must be presented in alphabetical order.

Sample Input

```
2
CBA
ED
CHEF
GECG
CHBIF
CBA
ED
IEGF
CHDA
CDADF
```

Sample Output

```
A 4 B 5 C 3 D 9 E 8 F 6 G 2 H 1 I 0
A 5 B 8 C 4 D 6 E 9 F 0 G 1 H 3 I 2
```

Problem B

Mutant Knights

Introduction

Chess players are well acquainted with the unorthodox movement of the knight. In its every move, this dodgy piece moves two squares in one direction and one square in a perpendicular direction – in other words, a knight's move is an L-shaped trajectory.

A well-known result in Mathematics states that a knight placed in any square in a standard 8×8 chessboard can reach any other square. Given this fact, we can define the *knight-distance* between two squares as the smallest number of knight moves necessary to go from one to the other (the knight-distance between a square and itself is 0). The figure below shows the knight-distance from every square to the bottom-left corner.

5	4	5	4	5	4	5	6
4	3	4	3	4	5	4	5
3	4	3	4	3	4	5	4
2	3	2	3	4	3	4	5
3	2	3	2	3	4	3	4
2	1	4	3	2	3	4	5
3	4	1	2	3	4	3	4
0	3	2	3	2	3	4	5

Problem

A *mutant knight* is a piece whose moves take it $m \geq 0$ squares in one direction and $n \geq 0$ squares in a perpendicular direction. Due to their specificity, mutant knights may not be capable of reaching any square from given squares in a chessboard. The goal of this problem is to determine the mutant-knight distance (defined as the knight distance) between two squares in a chess-like boards. A chess-like board is a rectangular board with $w \times h$ squares.

Input

The input consists of several lines, each one representing an instance of the problem. Each line contains eight numbers, corresponding (in this order) to:

- the parameters m and n of the mutant knight's move;
- the dimensions w and h of the board's width and height;
- the horizontal and vertical coordinates x_0 and y_0 of the origin square;
- the horizontal and vertical coordinates x_f and y_f of the destination square.

Assume that all numbers are between 0 and 999. The squares are numbered from left to right, bottom to top, counting from 1.

Output

The output should contain, for each input line, a single line containing the mutant-knight distance between squares (x_0, y_0) and (x_f, y_f) . If square (x_f, y_f) cannot be reached, then the line should contain the word `impossible`.

Sample Input

```
2 1 8 8 1 1 5 2
3 1 4 4 2 2 3 3
1 1 12 10 1 6 8 1
```

Sample Output

```
3
impossible
7
```

Problem C

Exam Scheduling

Introduction

For a given exam period, each student can register at most at one exam per course. When the exam registration finishes, it is time to schedule the exams in such a way that the number of students with consecutive exams is reduced to the minimum. By the number of students with consecutive exams we mean the number of students shared by two exams that are scheduled next to each other.

Problem

Write a program that given a list of student registrations, outputs for each pair of courses the minimum number of students with consecutive exams that can be obtained.

Given two courses A and B, the program has to find a sequence of courses beginning with A and ending with B that minimizes the total number of students with consecutive exams in that sequence. A student has a consecutive exam when he has a registration for two courses that are next to each other in the sequence.

For example, given the registration list $c1:s1, c2:s1, c3:s2$, where c_i represent courses and s_j represent students, the courses pairs $c1:c3$ and $c2:c3$ can have zero students with consecutive exams since they do not share students, but $c1:c2$ can also have zero students with consecutive exams by adding $c3$ between them, i.e. the sequence of courses $\langle c1, c3, c2 \rangle$ has no students with consecutive exams.

Thus the length of each sequence can vary from just two courses, A and B, to all the given courses. The sequences are independent between the pairs of courses, i.e. a sequence of courses found for one pair does not need to be maintained by other pairs.

Input

The input has several lines. Each line represents an independent instance of the problem. Each line (or instance) is composed by a list of courses in alphabetic order and separated by semicolons. For each course, we have a pair *course:students* where *course* is the course identifier and *students* is a list of students, in alphabetic order and separated by commas, which are registered in the *course*. The maximum number of courses is 100 and the maximum number of students is 200. Both courses and students are identified by a string with maximum length of 10.

Output

For each line in the input, the output of your program should consist of a line containing a list of all possible course pairs together with their minimum number of students with consecutive exams. The list has to be in alphabetic order, without the reflexive and symmetric pairs, and separated by semicolons. The course identifications and the minimum number are separated by colons.

Sample Input

c1:s1;c2:s1;c3:s2
c1:s1;c2:s1,s3;c3:s2;c4:s1,s2,s3
c1:s1,s3;c2:s1,s3;c3:s2,s3;c4:s1,s2,s3
c1:s1,s3;c2:s1,s3;c3:s2,s3;c4:s1,s2,s3;c5:s1,s4

Sample Output

c1:c2:0,c1:c3:0,c2:c3:0
c1:c2:0,c1:c3:0,c1:c4:1,c2:c3:0,c2:c4:1,c3:c4:1
c1:c2:2,c1:c3:1,c1:c4:2,c2:c3:1,c2:c4:2,c3:c4:2
c1:c2:2,c1:c3:1,c1:c4:2,c1:c5:1,c2:c3:1,c2:c4:2,c2:c5:1,c3:c4:1,c3:c5:0,c4:c5:1

Problem D

Gesture Recognition

Introduction

Your company decided on approaching a new market segment by introducing gesture recognition as an innovative technology. Gesture recognition has been a topic mainly explored in the artificial intelligence and the pattern recognition fields. The resulting solutions have been unable to reach an efficiency level in consonance with the demands of a daily use in resource constrained devices, like Smartphones.

Your company research seems, however, to have made a breakthrough, by finding an algorithm capable of robustly recognizing unistroke gestures. Moreover, this algorithm is invariant to the gesture's number of points, orientation and size.

The algorithm matches gestures performed by users to gesture templates previously specified. In order to be able to match gestures made with different number of points, orientation and size, the algorithm processes each gesture and template according to the following four steps:

1. Resample the gesture's points to a path with n evenly spaced points.
2. Rotate the points so that their indicative angle is 0° .
3. Scale the points so that the resulting gesture's bounding box is of $size^2$ dimension.
4. Translate the points so that the gesture's geometric center matches the origin.

The result of this processing is that all gestures have the same number of points, are rotated to have the same orientation, are bounded to a box of the same size, and have been translated so that their geometric center is at the same point.

Problem

You have been assigned the task of validating the proposed algorithm results. For this you will be provided with a set of templates which have been previously processed according to the presented algorithm. Additionally, you will receive a set of gestures to be classified, also already processed. All templates and gestures were resampled to 64 points, that is $n = 64$. You also have been assured that there will be no more than 20 templates in a single test.

A template is represented by a semicolon separated list of values. The first value is the template's name. The 128 remaining values represent the (x, y) coordinates of the 64 points comprising the template's gesture. For example:

```
arrow;-158;0;-150;-4;-142;-5;-134;-8;-126;-10;-118;-6;-110;-7;
-102;-7;-94;-5;-85;-5;-77;-6;-69;-5;-61;-3;-53;-5;-45;-1;-37;
-1;-29;4;-21;10;-12;15;-4;14;4;14;12;19;20;20;28;25;36;29;44;
28;52;25;58;12;54;-11;47;-25;40;-39;33;-52;25;-65;18;-77;10;
-86;2;-92;-5;-99;-9;-103;-1;-96;7;-89;15;-81;23;-73;30;-64;38;
-58;46;-51;54;-42;62;-34;70;-28;77;-18;85;-6;92;8;88;27;81;38;
```

74;50;66;57;58;68;51;79;43;90;36;101;28;108;20;118;13;127;5;
135;-2;147

is the template for the arrow gesture presented in figure 1.

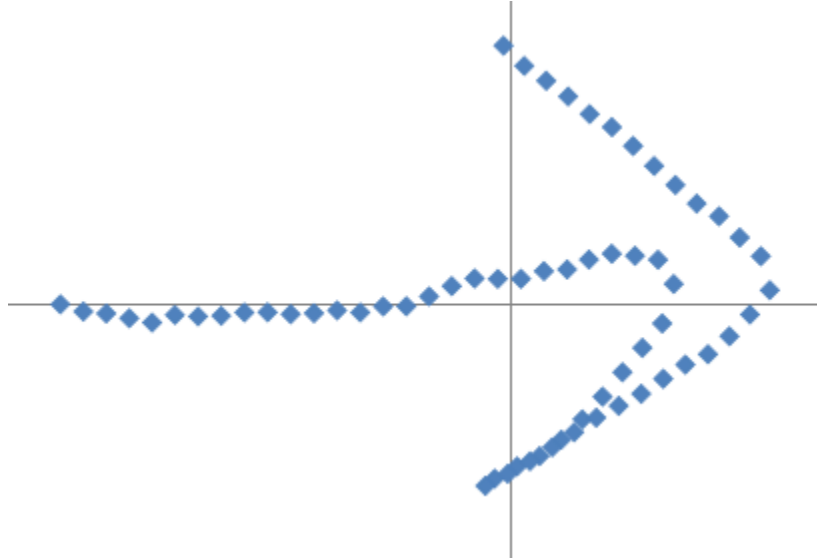


Figure 1: The arrow gesture

A gesture follows the same representation, without the name value.

It is your job to devise a method to perform the matching between gesture and template.

Input

The validation program starts by accepting templates. Each input line corresponds to one template. Template adding finishes when a *stop* command is issued. After this, the program enters recognition mode. In this mode, each line of the input is one gesture to be recognized. Recognition continues until a new *stop* command is issued.

Output

For each gesture input, the program must output the name of the template that most closely matches the gesture.

Sample Input

```
arrow;-158;0;-150;-4;-142;-5;-134;-8;-126;-10;-118;-6;-110;-7;
-102;-7;-94;-5;-85;-5;-77;-6;-69;-5;-61;-3;-53;-5;-45;-1;-37;
-1;-29;4;-21;10;-12;15;-4;14;4;14;12;19;20;20;28;25;36;29;44;
28;52;25;58;12;54;-11;47;-25;40;-39;33;-52;25;-65;18;-77;10;
-86;2;-92;-5;-99;-9;-103;-1;-96;7;-89;15;-81;23;-73;30;-64;38;
-58;46;-51;54;-42;62;-34;70;-28;77;-18;85;-6;92;8;88;27;81;38;
74;50;66;57;58;68;51;79;43;90;36;101;28;108;20;118;13;127;5;
135;-2;147
```

rectangle;-113;0;-108;10;-101;20;-96;30;-90;41;-85;51;-78;61;
 -73;71;-68;81;-61;91;-56;101;-49;111;-43;120;-34;119;-25;115;
 -16;108;-7;104;4;100;14;96;23;90;33;86;43;81;53;76;62;71;71;
 64;80;57;88;50;97;42;106;36;113;27;122;24;116;18;109;9;103;-1;
 97;-11;90;-20;84;-30;79;-41;73;-51;68;-62;63;-72;57;-82;51;
 -92;44;-101;38;-111;31;-120;25;-130;18;-122;10;-114;1;-109;-9;
 -103;-18;-97;-26;-90;-36;-84;-45;-78;-55;-72;-64;-67;-74;-62;
 -84;-57;-93;-50;-102;-43;-110;-36;-119;-29;-128;-23
 circle;-103;0;-108;11;-111;23;-111;35;-108;47;-102;58;-95;68;
 -86;77;-76;85;-65;92;-54;99;-43;105;-31;111;-19;116;-6;121;7;
 123;20;126;33;126;46;125;59;123;72;119;84;114;93;105;100;95;
 107;84;113;73;116;62;118;50;119;37;119;25;120;13;120;1;120;
 -11;117;-23;114;-35;110;-46;105;-58;100;-69;94;-80;86;-90;77;
 -99;68;-108;57;-114;45;-120;32;-122;19;-124;6;-123;-7;-120;
 -20;-116;-33;-113;-45;-109;-58;-105;-70;-101;-83;-96;-95;-91;
 -105;-84;-114;-75;-123;-66;-127;-54;-130;-42;-124;-31;-120;
 -20;-116;-8;-109;2
 triangle;-163;0;-151;7;-138;12;-125;18;-112;23;-98;28;-86;34;
 -73;40;-61;46;-49;53;-38;60;-26;67;-17;75;-6;83;4;91;15;98;25;
 106;32;114;43;122;52;131;57;125;58;115;59;105;63;95;66;84;69;
 74;70;63;72;53;74;43;75;32;75;21;76;11;76;0;75;-10;73;-21;72;
 -31;74;-41;75;-52;77;-62;80;-73;83;-83;85;-93;87;-104;85;-112;
 80;-119;66;-119;51;-116;39;-110;26;-104;13;-98;0;-93;-13;-87;
 -25;-80;-37;-74;-49;-67;-61;-60;-72;-54;-84;-47;-95;-39;-106;
 -32;-117;-24;-127;-16;-136;-8;-140;2
 stop
 -120;0;-113;9;-106;18;-98;26;-91;35;-84;44;-77;53;-71;62;-63;
 71;-56;80;-49;89;-42;97;-34;106;-27;115;-20;124;-12;127;-3;
 119;5;112;14;104;22;96;29;88;38;80;47;73;55;66;64;58;72;51;81;
 43;90;36;98;29;107;21;115;14;124;6;122;-2;115;-11;108;-20;100;
 -28;93;-37;86;-46;79;-55;71;-64;64;-72;57;-81;49;-89;42;-98;
 34;-107;27;-114;21;-122;13;-123;4;-116;-5;-110;-14;-102;-22;
 -95;-31;-87;-40;-80;-48;-73;-57;-65;-65;-58;-74;-50;-83;-43;
 -91;-35;-100;-28;-108;-21;-117;-14;-126;-7
 -160;0;-149;7;-136;13;-123;20;-113;28;-102;36;-89;41;-79;50;
 -67;57;-55;62;-43;69;-33;77;-21;84;-8;88;5;93;18;96;31;101;43;
 106;54;110;67;114;77;120;82;116;83;103;84;91;84;79;85;67;85;
 55;86;43;87;31;87;19;87;7;87;-5;87;-18;87;-30;87;-42;88;-54;
 88;-66;88;-78;88;-90;88;-102;86;-114;86;-126;81;-130;68;-124;
 55;-118;43;-112;30;-105;18;-98;5;-92;-6;-84;-17;-76;-28;-68;
 -37;-58;-48;-50;-59;-43;-69;-36;-80;-30;-91;-23;-103;-15;-114;
 -7;-127;-3;-137;4;-150;7;-162;7
 -162;0;-153;0;-145;-1;-136;-2;-128;-5;-119;-5;-110;-5;-101;-6;
 -93;-6;-84;-7;-75;-7;-67;-8;-58;-8;-49;-8;-40;-9;-32;-9;-23;
 -10;-14;-10;-6;-11;3;-16;11;-17;20;-17;29;-18;37;-18;46;-19;
 55;-19;54;-27;48;-39;41;-50;32;-55;25;-62;17;-69;10;-76;3;-83;
 -4;-92;0;-85;8;-78;16;-71;25;-66;33;-61;41;-55;49;-47;57;-40;

64;-30;70;-28;77;-21;84;-12;88;-2;87;7;80;19;73;31;67;42;60;
52;52;62;45;72;38;82;30;89;23;101;16;111;9;121;4;136;-2;148;
-9;158;-17;158
-99;0;-102;12;-97;23;-91;34;-86;46;-83;58;-79;69;-74;79;-68;89;
-58;97;-50;105;-40;110;-35;119;-21;123;-8;125;6;126;20;123;32;
119;45;116;57;111;70;106;80;99;91;93;102;85;109;75;112;63;116;
51;120;39;124;27;126;15;127;3;126;-9;125;-21;119;-32;114;-43;
108;-54;100;-64;93;-75;85;-85;78;-95;69;-105;56;-109;44;-114;
31;-118;17;-121;4;-124;-9;-122;-22;-122;-35;-119;-45;-116;-56;
-111;-68;-105;-78;-98;-87;-88;-94;-78;-102;-68;-107;-57;-111;
-45;-115;-34;-119;-22;-123;-10;-120;1;-114;10;-115;14
stop

Sample Output

rectangle
triangle
arrow
circle

Problem E

TIUPix's Electronic Invoices

Introduction

TIUPix is the newest and hottest mobile communications operator working in TIUPland and it's starting to gain momentum by increasing its market share.

Despite its huge success, the number of complaints keeps rising exponentially and the market controller is threatening to fine them harshly in a couple million TIUP dollars which could mean the end of the company.

Motive: Errors in their monthly electronic invoices due to the Happy Hours!

Problem

Your mission, should you choose to accept it, is saving TIUPix by creating a new algorithm to generate and calculate the invoice's total.

Take notice:

- There are three kinds of debits: VOI (voice), DAT (data) and SMS (Short Messages).
- Only Voice and Data consumptions are free during Happy Hours
- Happy Hours start each day at 21h00 (9:00 PM) and end at 09h00 (09:00 AM) the following day.
- A consumption that is only partially covered by a Happy Hour, only charges the consumption made outside of the Happy Hour.
- Both voice and data calls are charged by the second. This is, if I have a 67 seconds call (voice or data), I'll be charged for 67 seconds and not for two minutes.
- Calls can't be longer than 10 hours.
- Overlapped calls are allowed.
- The invoice only refers to a single month ($1 \leq \text{day} \leq 31$).
- TIUP dollars have a very, very low value when compared to other currencies like euros or US dollars. All numeric values used are integers.

Input

One line with the number of test cases ($1 \leq \text{tests_cases} \leq 100$).

One line with three values: cost for voice calls per minute, cost for data calls per second, cost per sms sent.

One line with the number of consumptions ($1 \leq n \leq 100$).

Following n lines with consumptions ordered by day with the following format:

```
<day> <VOI|DAT|SMS> <start time HH MM SS> <duration in seconds>
```

Output

First line should be "Invoice #XX", where XX stands for the test case number.

Output must be ordered by day, then by VOI, DAT and SMS, and should have the following format:

```
<day> <VOI|DAT|SMS> <number of charged seconds or number of SMS's sent> <cost>
```

Kinds of service that in a specific day don't have consumptions shouldn't be printed.

End with six lines where XX is the corresponding value:

```
-----
Total VOI: XX TIUP dollars
Total DAT: XX TIUP dollars
Total SMS: XX TIUP dollars
-----
Grand Total: XX TIUP dollars
```

After each test case output, including the last one, should be a blank line.

Sample Input

```
2
180 2 8
10
1 SMS 9 08 1 0
1 SMS 10 00 1 0
1 SMS 10 01 0 0
1 VOI 9 48 1 3600
1 DAT 10 08 1 1200
2 VOI 8 48 1 3600
2 SMS 12 08 1 0
2 DAT 18 08 1 10800
2 SMS 20 00 1 0
2 SMS 21 04 54 0
600 5 2
20
1 SMS 9 08 1 0
1 SMS 10 00 1 0
1 SMS 10 01 0 0
1 VOI 9 48 1 3600
1 DAT 10 08 1 1200
2 VOI 8 48 1 3600
2 SMS 12 08 1 0
2 DAT 18 08 1 10800
2 SMS 20 00 1 0
2 SMS 21 04 54 0
```

3 SMS 9 08 1 0
3 SMS 10 00 1 0
3 SMS 10 01 0 0
4 VOI 9 48 1 3600
5 DAT 10 08 1 1200
5 VOI 8 48 1 3600
31 SMS 12 08 1 0
31 DAT 18 08 1 10800
31 SMS 20 00 1 0
31 SMS 21 04 54 0

Sample Output

Invoice #1

1 VOI 3600 10800
1 DAT 1200 2400
1 SMS 3 24
2 VOI 2881 8643
2 DAT 10319 20638
2 SMS 3 24

Total VOI: 19443 TIUP dollars
Total DAT: 23038 TIUP dollars
Total SMS: 48 TIUP dollars

Grand Total: 42529 TIUP dollars

Invoice #2

1 VOI 3600 36000
1 DAT 1200 6000
1 SMS 3 6
2 VOI 2881 28810
2 DAT 10319 51595
2 SMS 3 6
3 SMS 3 6
4 VOI 3600 36000
5 VOI 2881 28810
5 DAT 1200 6000
31 DAT 10319 51595
31 SMS 3 6

Total VOI: 129620 TIUP dollars
Total DAT: 115190 TIUP dollars
Total SMS: 24 TIUP dollars

Grand Total: 244834 TIUP dollars