COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Code Warrior Introduction

Task #**1**	Novice	**1** Points

Task Description

Computer programmers must learn some rudimentary social skills, even if their interpersonal interaction is constrained by digital media. You should write a program to introduce your team to the judges.

Program Input

Your program should print a prompt for the judge to enter his or her name like the **bold face type** below. Then it should read the name from the keyboard input. You may assume that the judge will enter first name only, so you don't have to worry about spaces in the input.

Please enter your name: Moldwart

Program Output

After reading the judge's name, your program must complete the inroduction by (1) greeting the judge, and (2) displaying the name of your school, your mascot, and the names of each team member. You may use the format below as an example:

```
Hello, Moldwart!
We are Athens High School Lions: Mike Davis, Anh Nguyen, and Isok Patel.
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Secret Code

Task #2	Novice	2 Points

Task Description

Compaq has just recently discovered that a hacker has gained access to the mail traffic on its internal network. To prevent further unauthorized access to its interoffice email, Compaq has created a secret code that is applied to the text of all email messages. Your job is to write a program to change the line of text entered by the user using this new secret code.

The secret code replaces each vowel with a secret character as follows:

Vowel	Replacement
A	*
E	$
I	#
O	!
U	%
Y	^

Program Input

A line of text. End the program when the text “end” is entered.

```
Hewlett-Packard Company, Year 2000 Financial Report
end
```

Program Output

```
C!mp*q C!mp%t$r C!rp!r*t#!n, ^$*r 2000 F#n*nc#*l R$p!rt
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Calendar

Task #3	Novice	3 Points

Task Description

Write a program to print out a calendar for a particular month given the day on which the first of the month occurs together with the number of days in the month.

Program Input

Your program must prompt the user to input an integer representing the day of the week on which the month begins (1 for Sunday, 2 for Monday, ... , 7 for Saturday), and an integer specifying the number of days in the month (between 28 and 31 inclusive). Your user prompts must match the prompts below. You can assume that all input data will be valid.

```
Enter day:
3
Enter the number of days in the month:
30
```

Program Output

Your program should print the appropriate calendar for the month. You must print the calendar heading (Sun, Mon, …., Sat) as shown below.

```
Sun Mon Tue Wed Thr Fri Sat
            1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30
```

COMPAQ code wars IV

PROGRAMMING COMPETITION

Anagram

Task #4	Novice	4 Points

Task Description

An anagram is a word or phrase obtained by rearranging the letters of another word or phrase. For example “Neo” is an anagram of “One”, “schedule” is an anagram of “held cues”, and “never diets” is an anagram of “desert vine”. Write a program which can determine if two strings are anagrams.

Program Input

The input file consists of pairs of strings delimited by double-quote marks. The first line of the file specifies the number of string pairs. You may assume each string is no longer than 80 characters.

```
5
"one" "neo"
"sacred cow" "code wars"
"A stitch in time saves nine." "Is this meant as incentive?"
"cow adviser" "code wars iv"
"dormitory" "dirty room"
```

Program Output

The program must write each pair of strings to the output file and indicate if they are anagrams. Ignore punctuation, digits, and all other non-letters. Ignore the case of letters (as in the examples above). The program must write:

"string1" is an anagram of "string2"

if they are anagrams, or

"string1" is NOT an anagram of "string2"

otherwise.

```
"one" is an anagram of "neo"
"sacred cow" is NOT an anagram of "code wars"
"A stitch in time saves nine." is an anagram of "Is this meant as incentive?"
"cow adviser" is an anagram of "code wars iv"
"dormitory" is an anagram of "dirty room"
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Golf

Task #5	Novice	5 Points

Task Description

Tiger the Robot plays a perfect game of golf. When he hits the golf ball it always goes directly towards the hole on the green, and he always hits exactly the distance that is specified for the club. Each such action is known as a stroke, and the object of golf is to hit the ball from the tee to the hole in the fewest number of strokes. Tiger needs a program to select the best combination of clubs to reach the hole in the fewest strokes. Tiger can carry up to 14 clubs, and the total distance from the tee to any hole ranges from 100 to 600 yards. Tiger can also play up to 18 holes with the same set of clubs. Tiger must hit the ball the exact distance to the hole in order for the ball to drop in the hole.

Program Input

The first line of input is the number of clubs (between 1 and 14). For each club, a line follows specifying the distance (in yards) the club will hit the ball (between 1 and 300). The next line of input is the number of holes (between 1 and 18). For each hole, a line follows specifying the distance (in yards) from the tee to the hole (between 100 and 600) and the par for the hole (between 1 and 5). The notes to the right of the data will not be in the input file. They are provided here for clarification.

```
10     <- 10 clubs are in Tiger's bag
250    <- Yardage club 1 can hit the ball
175    <- Yardage club 2 can hit the ball
125    <- Yardage club 3 can hit the ball
75     <- Yardage club 4 can hit the ball
50     <- Yardage club 5 can hit the ball
25     <- Yardage club 6 can hit the ball
10     <- Yardage club 7 can hit the ball
5      <- Yardage club 8 can hit the ball
2      <- Yardage club 9 can hit the ball
1      <- Yardage club 10 can hit the ball
3      <- 3 holes for this golf course
453 5  <- First hole is 403 yards, par 5
133 3  <- Second hole is 133 yards, par 3
350 4  <- Third hole is 350 yards, par 4
```

Program Output

Print out Tiger's scorecard for this round of golf. For each hole, print the hole number, the minimum number of strokes for the hole, the number of strokes above or below par for the hole, the total number of strokes to this point, and the total number of strokes above or below par to this point (see below for an example). Output the results to an output file. You must include the scorecard header as shown below.

```
                        Total      Total
Hole    Strokes    +/-  Strokes    +/-
====    =======    ===  =======    =====
1       5          0    5          0
2       4          +1   9          +1
3       3          -1   12         0
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Particle Trajectory

Task #6	Novice/Advanced	6 Points

Task Description

A trajectory is a curve that describes the motion of an object, such as the orbit of a comet or the flight path of a rocket. Closer to the earth, an object that is thrust into the air, like a golf ball or artillery shell, also follows a curved motion from launch to impact. **Figure 1** is an example trajectory graph showing height and distance travelled.

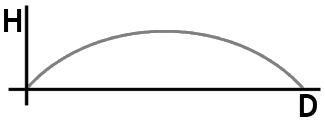


Figure 1

Your program will read the initial speed and launch angle from a file and then compute the flight time of an object along its trajectory, and also compute both the maximum vertical height and the horizontal distance travelled. For contest purposes we assume that (1) the only external force applied during the object's flight is gravity, and (2) the ground is completely flat.

Program Input

The first line of the input file specifies the number of trajectories described in the file. Each line thereafter is a separate trajectory descriptor. Each trajectory is described by two values: (1) the initial speed in meters per second, and (2) the launch angle in degrees.

```
2
44.8 30.0
72.1 56.9
```

Program Output

The program must write the three computed values to the output file using the text prompts shown in the example output below. Each trajectory should produce four lines of output in the order shown. You must include unit measures (like meters and seconds). For contest purposes you may ignore round-off error.

```
trajectory 1
horizontal distance: 177 meters
flight time: 4.57 seconds
maximum height: 25.6 meters
trajectory 2
horizontal distance: 485 meters
flight time: 12.3 seconds
maximum height: 186 meters
```

Hint

To solve this problem you'll need to know some specific math and physics. Since this is a programming contest and not a physics contest, we'll provide most of the basic information you should need to solve the problem.

To compute the horizontal (or X axis) and vertical (or Y axis) components of the initial velocity use the formulas below. See **Figure 2** for the corresponding vector diagram.

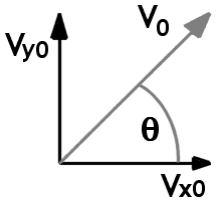


Figure 2

$$V_{x0} = V_0 \cos\theta \qquad V_{y0} = V_0 \sin\theta$$

You should also use the formula for linear distance $d=\frac{1}{2}at^2+v_0t$, linear velocity $v=at+v_0$, and the gravitational acceleration constant $a=-9.8 m/s^2$. The variables in these formulas are: a=acceleration, v=velocity*, t=time, d=distance.

One way to solve this task is to break it down into sub tasks like this:
1. Compute the time required to acheive maximum height by solving the linear velocity equation where $v_y=a_yt+v_{y0}=0$
2. Multiply the result of Step 1 by two to determine the total flight time.
3. Use the result of Step 1 in the linear distance formula to compute the maximum vertical height.

4. Use the result of Step 2 in the linear distance formula to compute the horizontal distance travelled.

* To be precise we are using the variable v as speed and not velocity *per se*. Vector math is beyond the scope of this task presentation.

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Simple Database

Task #7	Novice/Advanced	7 Points

Task Description

Cheap secondary storage (like hard drives and CD-ROMs) have encouraged the widespread use of computer databases. Databases are used in a wide variety of applications including ecommerce, factory automation, accounting, and company personnel records. Your task is to write a simple storage and query system.

Program Input

The input file consists of a header, a data section, and a query section. The first line of the header contains three integers: the number of data fields, the number of data records, and the number of queries. The second line of the header lists each field name. Field names are separated by commas. The maximum number of fields is nine, the maximum number of data rows or queries is one thousand.

The data section contains has a single line for each data record. The data field values are separated by commas.

There is one query per line in the query section of the input file. Each query consists of a field name, an equal sign, and a key value. A very simple example input file is shown below.

```
5 6 2
LASTNAME, FIRSTNAME, TITLE, DEPARTMENT, SALARY
Carter, Steve, Admin. Assistant, Facilities, 2700
Lin, Kyle, Manager, North America Sales, 4200
Gonzales, Maria, Engineer, Manufacturing Support, 4100
Anderson, Greg, Maintenance Technician, Facilities, 2950
Carter, Linda, Heroine, Customer Relations, 6500
Mudhi, Tariq, Sales Specialist, North America Sales, 3200
LASTNAME = Carter
DEPARTMENT = Manufacturing Support
```

Program Output

The program must respond to each query by writing a result table to the output file. The table should show all data rows that match the query. The result table must have a column for each data field, and the columns must align left, including the column headings. Each result set must be separated by a blank line. See the results below for examples.

```
LASTNAME FIRSTNAME TITLE            DEPARTMENT             SALARY
Carter   Steve     Admin. Assistant Facilities             2700
Carter   Linda     Heroine          Customer Relations     6500

LASTNAME FIRSTNAME TITLE            DEPARTMENT             SALARY
Gonzales Maria     Engineer         Manufacturing Support  4100
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Encryption Analyst

Task #8	Novice/Advanced	8 Points

Task Description

You've been contracted by your country's Intelligence Agency to decode a message that was intercepted by an operative in a foreign country. Your mission 008, should you decide to accept it, is to write an algorithm that processes an input data file and determines the encryption key. This key will be an 8-bit value.

Program Input

The input file will be encrypted and will consist of a list of values each ranging from 0-255, separated by commas and/or Carriage Returns/Linefeeds. You will need to apply an encryption key value ranging from 0-255 to each byte of the message and search on the keyword '**Codewars4**' (ignore case). When you have encountered this keyword, then you will have your encryption key and be able to decode the message. Each byte will then represent standard ASCII characters (ie-65='A', 66='B', 67='C', etc.).

Decryption Algorithm

The operative who intercepted the message was also able to obtain the decryption algorithm. Luckily for your country, he was able to send it, along with the message, just before they disappeared. The algorithm adds the encryption key to the first byte, adds the encryption key plus 1 to the second byte, adds the encryption key plus 2 to the third byte, and so on. When the (encryption key plus increment) reaches 255, the next value will be 0. Each byte is then masked to obtain final values that range from 0-255.

```
138,165,174,173, 162,167,165, 86, 92,119,162,150,150,135,144,160,
160, 96, 82,100, 73,113,149, 156,134,151,140,145,143, 64,150,135,
137,136, 59,137,124,123,140,136, 53,122,133,129,126, 48,131,118,
114, 44, 89, 97, 55, 40, 73,127, 37, 55, 68, 79, 45, 32,118, 99,
29, 97, 115,106, 94, 91,107, 0, 91,105, 95, 94, 17, 85, 93, 98,
95, 77, 91, 87, 78, 86, 91, 6, 84, 74, 3, 86, 73, 69,255, 63,
65, 82, 64, 76, 76, 57, 73, 79, 3,222,221

Example: With an encryption key of 0xC3, do the following to the input data:
1. The first decrypted value will be (((0xC3 + 0) & 0xFF) + 138) & 0xFF) -> 0x4D ('M')
2. The second decrypted value will be (((0xC3 + 1) & 0xFF) + 165) & 0xFF) -> 0x69 ('i')
3. The third decrypted value will be (((0xC3 + 2) & 0xFF) + 174) & 0xFF) -> 0x73 ('s')
4. The forth decrypted value will be (((0xC3 + 3) & 0xFF) + 173) & 0xFF) -> 0x73 ('s')
5. And so on…
```

Program Output

Upon decryption, the output will be a simple text message that will save your country. Simply output the encryption key followed by the message as follows to an output file. Be sure to account for word wrap such that no line has more than 80 characters, and no words are split between lines.

```
Encryption Key = 0xC3
Mission 'CodeWars4': Invasion will occur from the NW. By 3AM, we expect
full entrapment of the adversary.
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Name That Cow

Task #9	Novice/Advanced	8 Points

Task Description

Among the large Texas cattle ranchers, it is customary to brand cows with serial numbers to please the Accounting Department. The ranch hands don't appreciate the advantage of this filing system and wish to call the members of their herd by a sonorous name rather than saying, "C'mon, #4734, get along."

Help them out by writing a program that will translate the brand serial number of a cow into possible names uniquely associated with that serial number. Since the ranch hands all have cellular saddle phones these days, use the standard Touch-Tone(R) telephone keypad mapping to get from numbers to letters (except for "Q" and "Z"):

```
2: A,B,C        5: J,K,L        8: T,U,V
3: D,E,F        6: M,N,O        9: W,X,Y
4: G,H,I        7: P,R,S
```

Acceptable names for the cows are provided via an input file, which contains a list of fewer than 5,000 acceptable cattle names (first letter capitalized, of course). Take a cow's brand number and report which of all the possible words to which that number maps are in the given dictionary.

For instance, the brand number 4734 produces all the following names:

< Isfi Gpeh ... Gpeg Gpdi Gpdh Gpei Gpdg ... Greg ...>

As it happens, the only one of these 81 names that is in the list of valid names is "Greg".

Write a program that repeatedly asks for the brand number of a cow and prints all the valid names that can be generated from that brand number; print "No matching names found" if no valid words match. Serial numbers can be as many as a 9 digits long. End the program, when the serial number `0' is entered.

Program Input

```
232
252473
727225
0
```

Program Output

```
Possible names for #232 are: Ada
Possible names for #252473 are: Blaise, Claire
Possible names for #727225 are: Pascal
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Mars Rover

Task #10	Novice/Advanced	10 Points

Task Description

Your team is assigned to write simple navigation code for a Mars PRM (Planetary Rover Module). The PRM must be able to execute these simple commands: MOVE, RIGHT, LEFT, and STOP. These commands are transmitted to the PRM from the ADV (Atmospheric Descent Vehicle) which carried the PRM to the Martian surface. The PRM must supply feedback status to the ADV after each command (see *Table 1*).

The martian surface is represented by a two dimensional grid of ASCII text characters summarized in *Table 2*.

COMMAND	STATUS
LEFT or RIGHT	COMPLETE
MOVE	SUCCESS or FAILURE
STOP	STOPPED

Table 1. Command Set and Return Status Codes

CHAR	KEY
.	flat ground
*	obstacle
A	start position
o	ore
#	crystal

Table 2. Grid Chars

Note: A move command may fail (at least partially) if the PRM detects an obstacle in its path. When such a failure occurs, the PRM must indicate the relative position of the obstacle during the feedback transmission to the ADV.

Program Input

The input file consists of three sections: a *header*, a *map*, and a *command sequence*. The header is a single line of text with two fields: (1) an integer map size Z (where Z < 20) and (2) a starting orientation string. The orientation string may be either NORTH, SOUTH, EAST or WEST, with NORTH being at the top of the screen (or file). This orientation string tells your program which direction the PRM is facing at the start of the exploration.

The map is a character array of Z lines with Z characters per line separated by spaces. The valid input map characters and their meanings are shown in *Table 2* above. The command sequence follows the format in the Task Description. The MOVE command is followed by an integer declaring the number of units which the PRM must move forward.

```
8 NORTH
. . . . . . . .
. . . . . . . .
. . . . . * .
. . . . . . . .
. * . . . . # .
. . o . . . . .
. . . . . . . .
. A . . * . . .
MOVE 3
RIGHT
MOVE 2
LEFT
MOVE 4
STOP
```

Program Output

The output file must consist of two sections: a *command feedback log* and a *result map*. The command feedback log must have a line of text for each command from the input file. The first part of the line echoes the command, and the second part reports the ADV feedback string. If ore or crystal is discovered on the path then the program must print a status message indicating the type of discovery and the position (see the example below). You may assume that the move commands will not put the PRM outside the map boundaries.

The result map must be printed according to these rules: (1) It must show the path taken by the PRM with plus

characters. (2) It must show the start position with an A and the final stopped position of the PRM with a capital P. (3) It must show obstacles that interfered with movement with a * character. (4) It must show ore and crystal discoveries with the appropriate # or o character. (5) All other positions must be written with a ? character.

Your program must write the command feedback log and result map with the same syntax and format as the sample below.

```
MOVE 3: FAILURE DUE TO OBSTACLE AT POSITION 3.
RIGHT: COMPLETE.
MOVE 2: SUCCESS.
ORE DISCOVERED AT POSITION 1.
LEFT: COMPLETE.
MOVE 4: SUCCESS.
STOP: STOPPED.
? ? ? ? ? ? ? ?
? ? ? P ? ? ? ?
? ? ? . ? ? ? ?
? ? ? . ? ? ? ?
? * ? . ? ? ? ?
? . o . ? ? ? ?
? . ? ? ? ? ? ?
? A ? ? ? ? ? ?
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Job Processing

Task #11	Advanced	11 Points

Task Description

A factory is running a production line. Two operations have to be performed on each job: first operation "A", then operation "B". There is a certain number of machines capable of performing each operation. **Figure 1** shows the organization of the production line that works as follows: A type "A" machine takes a job from the input container, performs operation "A" and puts the job into the intermediate container. A type "B" machine takes a job from the intermediate container, performs operation "B" and puts the job into the output container. All machines can work in parallel and independently of each other, and the size of each container is unlimited. The machines have different performance characteristics, so each machine works with a given processing time.

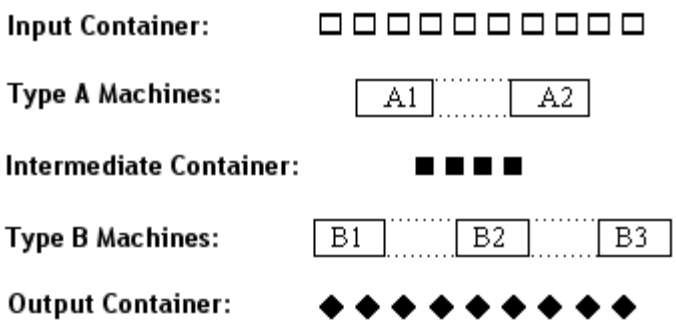


Figure 1

Your program must compute the ***minimal amount of time*** operation "A" can be completed for all N jobs provided that the jobs are available at time 0. It must also compute the ***minimal amount of time*** that is necessary to perform both operations on N jobs.

Source: 8th International Olympiad in Informatics, Veszprém, Hungary
http://www.inf.bme.hu/contests/tasks/ioi96d1.html

Program Input

The input file contains positive integers in five lines. The first line contains N, the number of jobs (1<=N<=1000). On the second line, the number M1 of type "A" machines (1<=M1<=30) is given. In the third line there are M1 integers, the job processing times of each type "A" machine. The fourth and the fifth line contain the number M2 of type "B" machines (1<=M2<=30) and the job processing times of each type "B" machine, respectively. The job processing time is measured in units of time, which includes the time needed for taking a job from a container before processing and putting it into a container after processing. Each processing time is at least 1 and at most 20.

```
5
2
1 1
3
3 1 4
```

Program Output

Your program should write two lines to the output file. The first line should contain the solution of subtask A. The second line should contain the solution of subtask B. The program must write the sentence labels as shown in the example below:

```
Minimum time to complete subtask A: 3
Minimum time to complete subtask B: 5
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Double Knockout Competition

Task #12	Advanced	12 Points

Task Description

In a number of sports, a championship may be determined by a double knockout competition. A team is eliminated on its second loss, so the winner is the last remaining team with one or fewer losses. The competition is played in a series of rounds: in each round, teams that have not been eliminated are paired subject to the constraint that an undefeated team never plays a team with one loss. As many teams as possible are paired in each round. After a number of rounds only two teams remain. These teams play in a round by themselves, although one is undefeated and the other is not. If neither is eliminated, they play again in a final round. For our analysis we assume that this extra round is always necessary.

Write a program to report on a Double Knockout Competition.

Program Input

The first line of the input file contains one positive integer n (which is the number of test cases that follow it). The next n lines each contain one positive integer t (where t < 32768), which is the number of teams in the competition for this test case.

```
5
2
6
32
64
512
```

Program Output

For each case print a summary of the competition to a file. The summary must detail each round as follows:
`Round 0: 2 undefeated, 0 one-loss, 0 eliminated, 0 games played`

This is followed by a similar line for each round of the competition, followed by a single line saying how my rounds and the total number of games played. The output for each different test case is to be separated by a single blank line.

```
Number of Teams: 2
Round: 0, 2 undefeated, 0 one-loss, 0 eliminated, 0 games played
Round: 1, 1 undefeated, 1 one-loss, 0 eliminated, 1 games played
Round: 2, 0 undefeated, 2 one-loss, 0 eliminated, 1 games played
Round: 3, 0 undefeated, 1 one-loss, 1 eliminated, 1 games played
There are 3 rounds and a total of 3 games were played.

Number of Teams: 6
Round: 0, 6 undefeated, 0 one-loss, 0 eliminated, 0 games played
Round: 1, 3 undefeated, 3 one-loss, 0 eliminated, 3 games played
Round: 2, 2 undefeated, 3 one-loss, 1 eliminated, 2 games played
Round: 3, 1 undefeated, 3 one-loss, 2 eliminated, 2 games played
Round: 4, 1 undefeated, 2 one-loss, 3 eliminated, 1 games played
Round: 5, 1 undefeated, 1 one-loss, 4 eliminated, 1 games played
Round: 6, 0 undefeated, 2 one-loss, 4 eliminated, 1 games played
Round: 7, 0 undefeated, 1 one-loss, 5 eliminated, 1 games played
There are 7 rounds and a total of 11 games were played.

Number of Teams: 32
Round: 0, 32 undefeated, 0 one-loss, 0 eliminated, 0 games played
Round: 1, 16 undefeated, 16 one-loss, 0 eliminated, 16 games played
Round: 2, 8 undefeated, 16 one-loss, 8 eliminated, 16 games played
Round: 3, 4 undefeated, 12 one-loss, 16 eliminated, 12 games played
Round: 4, 2 undefeated, 8 one-loss, 22 eliminated, 8 games played
Round: 5, 1 undefeated, 5 one-loss, 26 eliminated, 5 games played
Round: 6, 1 undefeated, 3 one-loss, 28 eliminated, 2 games played
Round: 7, 1 undefeated, 2 one-loss, 29 eliminated, 1 games played
Round: 8, 1 undefeated, 1 one-loss, 30 eliminated, 1 games played
```

```
Round: 9, 0 undefeated, 2 one-loss, 30 eliminated, 1 games played
Round: 10, 0 undefeated, 1 one-loss, 31 eliminated, 1 games played
There are 10 rounds and a total of 63 games were played.

Number of Teams: 64
Round: 0, 64 undefeated, 0 one-loss, 0 eliminated, 0 games played
Round: 1, 32 undefeated, 32 one-loss, 0 eliminated, 32 games played
Round: 2, 16 undefeated, 32 one-loss, 16 eliminated, 32 games played
Round: 3, 8 undefeated, 24 one-loss, 32 eliminated, 24 games played
Round: 4, 4 undefeated, 16 one-loss, 44 eliminated, 16 games played
Round: 5, 2 undefeated, 10 one-loss, 52 eliminated, 10 games played
Round: 6, 1 undefeated, 6 one-loss, 57 eliminated, 6 games played
Round: 7, 1 undefeated, 3 one-loss, 60 eliminated, 3 games played
Round: 8, 1 undefeated, 2 one-loss, 61 eliminated, 1 games played
Round: 9, 1 undefeated, 1 one-loss, 62 eliminated, 1 games played
Round: 10, 0 undefeated, 2 one-loss, 62 eliminated, 1 games played
Round: 11, 0 undefeated, 1 one-loss, 63 eliminated, 1 games played
There are 11 rounds and a total of 127 games were played.

Number of Teams: 512
Round: 0, 512 undefeated, 0 one-loss, 0 eliminated, 0 games played
Round: 1, 256 undefeated, 256 one-loss, 0 eliminated, 256 games played
Round: 2, 128 undefeated, 256 one-loss, 128 eliminated, 256 games played
Round: 3, 64 undefeated, 192 one-loss, 256 eliminated, 192 games played
Round: 4, 32 undefeated, 128 one-loss, 352 eliminated, 128 games played
Round: 5, 16 undefeated, 80 one-loss, 416 eliminated, 80 games played
Round: 6, 8 undefeated, 48 one-loss, 456 eliminated, 48 games played
Round: 7, 4 undefeated, 28 one-loss, 480 eliminated, 28 games played
Round: 8, 2 undefeated, 16 one-loss, 494 eliminated, 16 games played
Round: 9, 1 undefeated, 9 one-loss, 502 eliminated, 9 games played
Round: 10, 1 undefeated, 5 one-loss, 506 eliminated, 4 games played
Round: 11, 1 undefeated, 3 one-loss, 508 eliminated, 2 games played
Round: 12, 1 undefeated, 2 one-loss, 509 eliminated, 1 games played
Round: 13, 1 undefeated, 1 one-loss, 510 eliminated, 1 games played
Round: 14, 0 undefeated, 2 one-loss, 510 eliminated, 1 games played
Round: 15, 0 undefeated, 1 one-loss, 511 eliminated, 1 games played
There are 15 rounds and a total of 1023 games were played.
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Gene Sequencing

Task #13	Advanced	14 Points

Task Description

You are working on the human genome project, and your team is closing in on the last few base pairs needed to complete the project and go on to certain glory and perhaps a Nobel prize. Researchers on your team have managed to obtain several sets of segments needed to sequence the last remaining gene, but your help is urgently needed to lead the effort.

Your job is to write a program which can reconstruct the gene sequence using the given fragmentary segments of the larger gene. For each set of segments given, you must devise an algorithm to sequence them in the correct order. If you succeed, your name will appear in all biology textbooks from this day forward. If you are too slow, someone else will take your place. Good luck.

source: The MATLAB Programming Contest - http://www.mathworks.com/contest/sequence.cgi/rules.html

Program Input

The gene segments given to you have been chemically snipped from a single gene and sequenced individually. Your job is to reassemble them jigsaw-style into the most likely single sequence they could have come from. The gene segments will be represented in the input file as a character (or string) matrix.

Only four letters will be used. Each stands for one of the nitrogenous bases in DNA: A (adenine), C (cytosine), G (guanine), and T (thymine). The segments in any single test sequence have the same length (since they're passed as a char matrix). But that length, as well as the number of segments, will vary across the many tests included in our test suite - in other words, your program will be given inputs of different, unknown sizes. In one case, we might have 5 segments of length 4. Another test case might provide 10 segments of length 12. For our purposes we will limit all test cases to a maximum of 10 segments with a maximum segment length of 16.

To provide a sense of scale, the entire human genome is approximately three billion base pairs long.

The first line of the input file indicates the total number of segment sets. Each set begins with a line of two integers specifying the number of segments N and the segment size Z, respectively. The next N lines will be strings of length Z.

```
2
5 4
TCGG
GCAG
ATCG
CAGC
GATC
4 16
AGCTTGCTGAAACGTA
CTTGCTGAAACGTATC
TATCGATGCATCGATC
AAACGTATCGATGCAT
```

Program Output

Your program should determine the shortest gene that can be made from these pieces. The rules for assembling a sequence are: (1) the sequence must use all the segments (and only the segments) provided, (2) you cannot flip any of the segments, (3) the result must be a one row char array, (4) the best answer is the shortest answer, (5) speed counts. For example, if we slide around the above segments we can get:

```
  TCGG
     GCAG
 ATCG
       CAGC
GATC
==========
GATCGGCAGC
```

This solution has length 10. It should be obvious that there will be many solutions for each set. The easiest solution is

just to stick all the pieces placed end to end. While this sequence is valid, it's unlikely to be a faithful reconstruction of the original gene and would therefore be penalized because of its length. The team with the shortest solution will threrefor earn a **3 point bonus**. Keep in mind that length isn't the only criteria. Your entry will be judged also on its speed. The program must execute in less than 30 seconds on a K6 or Pentium at 200MHz or it will be marked incorrect. The program output file should show each solution on a separate line.

```
GATCGGCAGC
AGCTTGCTGAAACGTATCGATGCATCGATC
```

COMPAQ code wars IV

P R O G R A M M I N G C O M P E T I T I O N

Simplex Manufacturing

Task #14	Advanced	15 Points

Task Description

Computers can aided greatly in the analysis of critical business factors in order to make optimum business operations decisions. Your task is to determine the optimum product mix in order to achieve total optimum revenue each week for a given set of variable operational factors and variables market conditions for Compaq Computer Corporation. There is a given set of fixed factors that will be the same every week. The fixed factors are as follows:

- There are 40 worker-hours available per manufacturing employee to build product each week.
- It takes 3 hours for 1 manufacturing employee to assemble a Compaq ProLiant Server.
- It takes 1 hours for 1 manufacturing employee to assemble a Compaq Laptop computer.
- It takes 20 lbs. of sheet metal to build one Compaq ProLiant Server.
- It takes 2 lb. Of sheet metal to build one Compaq Laptop computer.
- It takes 1 lb. of plastic to build one Compaq ProLiant Server.
- It takes 3 lbs. of plastic to build one Compaq Laptop Computer.
- Demand for Compaq products is such that every server or laptop assembled is immediately considered sold and included in that week's revenue

The weekly fluctuating factors, which will be the Program Input are as follows:

- Number of Manufacturing employees available to work for a given week.
- Lbs. of plastic on hand for a given week.
- Lbs. of sheet metal available for a given week.
- Selling Price of a Compaq ProLiant Server.
- Selling Price of a Compaq Laptop Computer.

Program Input

The input file will contain 12 lines that correspond to 12 different weeks. Each line will consist of 5 numbers that are separated by commas: # of employees, # of Lbs. of Sheet metal available, # of Lbs. of plastic available, Server Selling Price, Laptop Selling Price.
For example the input on one line would be:

511, 4025, 3000, 6400, 2350

Program Output

The program must respond to each week's input line by generating one output line that has 3 numbers separated by commas: optimum # of servers to manufacture, optimum # of laptops to manufacture, Optimum Total Revenue generated (in dollars, rounded to the nearest dollar). Compaq can't ship partially assembled units so only count the # of whole units that can be manufactured when outputting the optimum #'s of servers and laptops. Since the

Program Input will have 12 lines, the Program Output should also have 12 lines.

The example single input line given above would generate the following output example:

104, 965, 2933350

This signifies that to achieve the optimum revenue for the given example week, Compaq manufacturing should assemble 104 servers and 965 laptops for total revenue of $2,933,350 dollars.

COMPAQ code wars IV

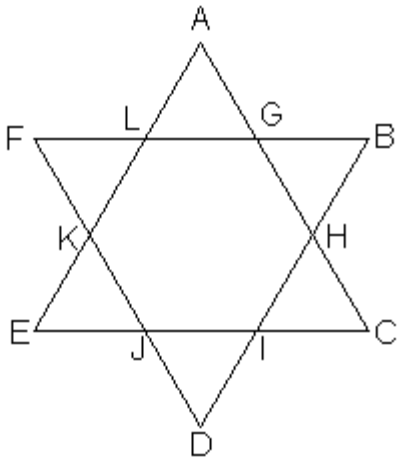
P R O G R A M M I N G C O M P E T I T I O N

Magic Star Numbers

Task #15	Advanced	18 Points

Task Description

This is a magic number puzzle. You are given a six-pointed star as shown in the illustration below. You must assign a unique value from 5 to 16 at each line intersection in the star, with A = 5. What number assignments result in **all sums** along any straight line equaling 42? For instance A + G + H + C = 42, C + I + J + E = 42 etc.



Your task is to write a program that will find all the answers to this puzzle. Mirror versions of an answer are considered a different answer.

Program Input

None.

Program Output

The program screen output should look like what follows. The program should list the title, a "." for every 100,000 possible solutions checked, each answer as it is found on a separate line, the number of solutions checked and the total number of answers found at the end of the program.

```
Magic Star Number Solutions

.....some number of periods.....

Solution 1: A=5 B=7 C=6 D=8 E=13 F=9 G=16 H=15 I=12 J=11 K=14 L=10

.....the rest of the solutions and periods.....

Total solutions checked = ??
Total solutions found = ??
```

The Program should also write the answers to a file, one solution per line, values separated by commas, with the solution # as the first item and the numbers for the solution in A to L order following, as shown below.

```
1,5,7,6,8,13,9,16,15,12,11,14,10
```