The Eighteenth Annual acm International Collegiate

Programming Contest Finals

sponsored by

Microsoft®

Problem A
Borrowers

> I mean your *borrowers of books*—those mutilators of collections, spoilers of the symmetry of shelves, and creators of odd volumes.
> —Charles Lamb, *Essays of Elia* (1823) 'The Two Races of Men'

Like Mr. Lamb, librarians have their problems with borrowers too. People don't put books back where they should. Instead, returned books are kept at the main desk until a librarian is free to replace them in the right places on the shelves. Even for librarians, putting the right book in the right place can be very time-consuming. But since many libraries are now computerized, you can write a program to help.

When a borrower takes out or returns a book, the computer keeps a record of the title. Periodically, the librarians will ask your program for a list of books that have been returned so the books can be returned to their correct places on the shelves. Before they are returned to the shelves, the returned books are sorted by author and then title using the ASCII collating sequence. Your program should output the list of returned books in the same order as they should appear on the shelves. For each book, your program should tell the librarian which book (including those previously shelved) is already on the shelf before which the returned book should go.

Input

First, the stock of the library will be listed, one book per line, in no particular order. Initially, they are all on the shelves. No two books have the same title. The format of each line will be:

`"`*title*`" by` *author*

The end of the stock listing will be marked by a line containing only the word:

`END`

Following the stock list will be a series of records of books borrowed and returned, and requests from librarians for assistance in restocking the shelves. Each record will appear on a single line, in one of the following formats:

`BORROW "`*title*`"`
`RETURN "`*title*`"`
`SHELVE`

The list will be terminated by a line containing only the word:

`END`

Output

Each time the `SHELVE` command appears, your program should output a series of instructions for the librarian, one per line, in the format:

`Put "`$title_1$`" after "`$title_2$`"`

or, for the special case of the book being the first in the collection:

`Put "`*title*`" first`

After the set of instructions for each `SHELVE`, output a line containing only the word:

`END`

Assumptions & Limitations

1. *A title is at most 80 characters long.*
2. *An author is at most 80 characters long.*
3. *A title will not contain the double quote (") character.*

Sample Input

```
"The Canterbury Tales" by Chaucer, G.
```

```
"Algorithms" by Sedgewick, R.
"The C Programming Language" by Kernighan, B. and Ritchie, D.
END
BORROW "Algorithms"
BORROW "The C Programming Language"
RETURN "Algorithms"
RETURN "The C Programming Language"
SHELVE
END
```

Output for the Sample Input

```
Put "The C Programming Language" after "The Canterbury Tales"
Put "Algorithms" after "The C Programming Language"
END
```

Problem B
Testing the CATCHER

A military contractor for the Department of Defense has just completed a series of preliminary tests for a new defensive missile called the CATCHER which is capable of intercepting multiple incoming offensive missiles. The CATCHER is supposed to be a remarkable defensive missile. It can move forward, laterally, and downward at very fast speeds, and it can intercept an offensive missile without being damaged. But it does have one major flaw. Although it can be fired to reach any initial elevation, it has no power to move higher than the last missile that it has intercepted.

The tests which the contractor completed were computer simulations of battlefield and hostile attack conditions. Since they were only preliminary, the simulations tested only the CATCHER's vertical movement capability. In each simulation, the CATCHER was fired at a sequence of offensive missiles which were incoming at fixed time intervals. The only information available to the CATCHER for each incoming missile was its height at the point it could be intercepted and where it appeared in the sequence of missiles. Each incoming missile for a test run is represented in the sequence only once.

The result of each test is reported as the sequence of incoming missiles and the total number of those missiles that are intercepted by the CATCHER in that test.

The General Accounting Office wants to be sure that the simulation test results submitted by the military contractor are attainable, given the constraints of the CATCHER. You must write a program that takes input data representing the pattern of incoming missiles for several different tests and outputs the maximum numbers of missiles that the CATCHER can intercept for those tests. For any incoming missile in a test, the CATCHER is able to intercept it if and only if it satisfies one of these two conditions:

1. The incoming missile is the first missile to be intercepted in this test.

-or-

2. The missile was fired after the last missile that was intercepted and it is not higher than the last missile which was intercepted.

Input and Output

The input data for any test consists of a sequence of one or more non-negative integers, all of which are less than or equal to 32,767, representing the heights of the incoming missiles (the test pattern). The last number in each sequence is `-1`, which signifies the end of data for that particular test and is not considered to represent a missile height. The end of data for the entire input is the number `-1` as the first value in a test; it is not considered to be a separate test.

Output for each test consists of a test number (Test #1, Test #2, etc.) and the maximum number of incoming missiles that the CATCHER could possibly intercept for the test. That maximum number appears after an identifying message. There must be at least one blank line between output for successive data sets. On the back of this page is a sample input file which consists of two different scenarios and the corresponding output.

NOTE: The number of missiles for any given test is not limited. If your solution is based on an inefficient algorithm, it **may not** execute in the allotted time.

Sample Input

```
389
207
155
300
299
170
158
65
-1
23
34
21
-1
-1
```

Output for the Sample Input

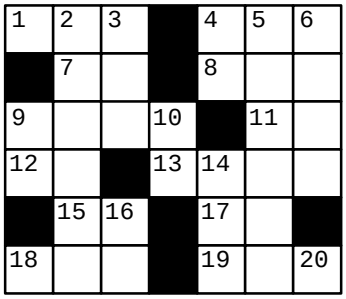
```
Test #1:
  maximum possible interceptions: 6

Test #2:
  maximum possible interceptions: 2
```

Problem C

Crossword Answers

A crossword puzzle consists of a rectangular grid of black and white squares and two lists of definitions (or descriptions). One list of definitions is for “words” to be written left to right across white squares in the rows and the other list is for words to be written down white squares in the columns. (A word is a sequence of alphabetic characters.) To solve a crossword puzzle, one writes the words corresponding to the definitions on the white squares of the grid.

The definitions correspond to the rectangular grid by means of sequential integers on “eligible” white squares. White squares with black squares immediately to the left or above them are "eligible." White squares with no squares either immediately to the left or above are also “eligible.” No other squares are numbered. All of the squares on the first row are numbered. The numbering starts with 1 and continues consecutively across white squares of the first row, then across the eligible white squares of the second row, then across the eligible white squares of the third row and so on across all of the rest of the rows of the puzzle. The picture below illustrates a rectangular crossword puzzle grid with appropriate numbering.



An “across” word for a definition is written on a sequence of white squares in a row starting on a numbered square that does not follow another white square in the same row. The sequence of white squares for that word goes across the row of the numbered square, ending immediately before the next black square in the row or in the rightmost square of the row.

A “down” word for a definition is written on a sequence of white squares in a column starting on a numbered square that does not follow another white square in the same column. The sequence of white squares for that word goes down the column of the numbered square, ending immediately before the next black square in the column or in the bottom square of the column. Every white square in a correctly solved puzzle contains a letter.

You must write a program that takes several solved crossword puzzles as input and outputs the lists of across and down words which constitute the solutions.

Input

Each puzzle solution in the input starts with a line containing two integers *r* and *c* (1• *r* •10 and 1• *c* •10), where *r* (the first number) is the number of rows in the puzzle and *c* (the second number) is the number of columns. The *r* rows of input which follow each contain *c* characters (excluding the end-of-line) which describe the solution. Each of those *c* characters is an alphabetic character which is part of a word or the character "*", which indicates a black square. The end of input is indicated by a line consisting of the single number 0.

Output

Output for each puzzle consists of an identifier for the puzzle (puzzle #1, puzzle #2, etc.) and the list of across words followed by the list of down words. Words in each list must be output one-per-line in increasing order of the number of their corresponding definitions. The heading for the list of across words is “Across”. The heading for the list of down words is “Down”. In the case where the lists are empty (all squares in the grid are black), the Across and Down headings should still appear.

Sample Input

```
2 2
AT
*O
6 7
AIM*DEN
*ME*ONE
UPON*TO
SO*ERIN
*SA*OR*
IES*DEA
0
```

Output for the Sample Input

```
puzzle #1:
Across
  1.AT
  3.O
Down
  1.A
  2.TO


puzzle #2:
Across
  1.AIM
  4.DEN
  7.ME
  8.ONE
  9.UPON
 11.TO
 12.SO
 13.ERIN
 15.SA
 17.OR
 18.IES
 19.DEA
Down
  1.A
  2.IMPOSE
  3.MEO
  4.DO
  5.ENTIRE
  6.NEON
  9.US
 10.NE
 14.ROD
 16.AS
 18.I
 20.A
```

Problem D

Package Pricing

The Green Earth Trading Company sells 4 different sizes of energy-efficient fluorescent light bulbs for use in home lighting fixtures. The light bulbs are expensive, but last much longer than ordinary incandescent light bulbs and require much less energy. To encourage customers to buy and use the energy-efficient light bulbs, the company catalogue lists special packages which contain a variety of sizes and numbers of the light bulbs. The price of a package is always substantially less than the total price of the individual bulbs in the package.

Customers typically want to buy several different sizes and numbers of bulbs. You are to write a program to determine the least expensive collection of packages that satisfy any customer's request.

Input

The input file is divided into two parts. The first one describes the packages which are listed in the catalogue. The second part describes individual customer requests. The 4 sizes of light bulbs are identified in the input file by the characters `"a"`, `"b"`, `"c"`, and `"d"`.

The first part of the input file begins with an integer n ($1 \bullet n \bullet 50$) indicating the number of packages described in the catalogue. Each of the n lines that follows is a single package description. A package description begins with a catalogue number (a positive integer) followed by a price (a real number), and then the sizes and corresponding numbers of the light bulbs in the package. Between 1 and 4 different sizes of light bulbs will be listed in each description. The listing format for these size-number pairs is a blank, a character (`"a"`, `"b"`, `"c"`, or `"d"`) representing a size, another blank, and then an integer representing the number of light bulbs of that size in the package. These size-number pairs will not appear in any particular order, and there will be no duplicate sizes listed in any package. The following line describes a package with catalogue number 210 and price $76.95 which contains 3 size `"a"` bulbs, 1 size `"c"` bulb, and 4 size `"d"` bulbs.

```
210 76.95 a 3 c 1 d 4
```

The second part of the input file begins with a line containing a single positive integer m representing the number of customer requests. Each of the remaining m lines is a customer request. A listing of sizes and corresponding numbers of light bulbs constitutes a request. Each list contains only the size-number pairs, formatted the same way that the size-number pairs are formatted in the catalogue descriptions. Unlike the catalogue descriptions, however, a customer request may contain duplicate sizes. The following line represents a customer request for 1 size `"a"` bulb, 2 size `"b"` bulbs, 2 size `"c"` bulbs, and 5 size `"d"` bulbs.

```
a 1 d 5 b 1 c 2 b 1
```

Output

For each request, print the customer number (1 through m, 1 for the first customer request, 2 for the second, …, m for the m^{th} customer), a colon, the total price of the packages which constitute the least expensive way to fill the request, and then the combination of packages that the customer should order to fill that request.

Prices should be shown with exactly two significant digits to the right of the decimal. The combination of packages must be written in ascending order of catalogue numbers. If more than one of the same type package is to be ordered, then the number ordered should follow the catalogue number in parentheses. You may assume that each customer request can be filled. In some cases, the least expensive way to fill a customer request may contain more light bulbs of some sizes than necessary to fill the actual request. This is acceptable. What matters is that the customers receive *at least* what they request.

Sample Input

```
5
10 25.00 b 2
502 17.95 a 1
3 13.00 c 1
55 27.50 b 1 d 2 c 1
6 52.87 a 2 b 1 d 1 c 3
6
d 1
b 3
b 3 c 2
b 1 a 1 c 1 d 1 a 1
b 1 b 2 c 3 c 1 a 1 d 1
b 3 c 2 d 1 c 1 d 2 a 1
```

Output for the Sample Input

```
1:   27.50 55
2:   50.00 10(2)
3:   65.50 3 10 55
4:   52.87 6
5:   90.87 3 6 10
6:  100.45 55(3) 502
```

Problem E

Switching Channels

CPN (The Couch Potato Network) owns several cable channels. They would like to arrange the timing of programmes so viewers can switch channels without missing the end of one programme or the beginning of another. To do this they have identified certain times, called “alignment points,” where ideally one programme should end and another should begin. Some of these alignment points are more important than others. For example, the time when the nightly news begins is an important alignment point. Since many viewers watch the news, they would be less likely to watch a CPN programme whose ending time causes them to miss the beginning of the news, or which starts before the news finishes. Your task is to write a solution which determines the best order in which programmes can be shown on one channel.

A “miss” time is the absolute value of the difference between the time of an alignment point and the nearest time of the beginning or end of a programme. The “total miss time” at a particular level of importance is the sum of all the miss times for alignment points at that level of importance. One programme order is better than another if it has a lower total miss time at some level of importance and the same total miss time at all higher levels of importance (if any).

Input

Your solution must accept multiple input data sets. Each set will begin with an integer, p ($0 \bullet p \bullet 8$), specifying the number of programmes to be ordered. When a data set beginning with 0 is encountered, your solution should terminate. Following p on the same line will be p integers specifying the lengths of the programmes in minutes. There is no significance to the order in which these are given.

The next line of input specifies the alignment points. The total number of such points, a ($0 \bullet a \bullet 8$), appears first followed by a pairs of integers. The first integer in each pair, i ($1 \bullet i \bullet 5$), gives the importance of the alignment point. Alignment points marked 1 are most important; those marked 2 are of secondary importance, etc. The second integer in each pair, t, specifies the time when the alignment point occurs. No two alignment points in the same data set will have the same value of t.

Output

Your solution must output three lines for each data set. The first line identifies the data set being processed and should be in the form:

```
Data set n
```

where n is the number of the data set (1 for the first data set, 2 for the second, etc.). On the following line, your solution should output the lengths of the programmes in the order in which they should be shown to achieve the best synchronization with the alignment points. On the third line, output the total number of minutes by which the alignment points were missed (the sum of all total miss times).

There may be more than one best programme order for an input data set. Any one of these best orders is acceptable.

Sample Input

```
4  30 45 45 15
3  1 60  2 90  3 15
6  10 15 13 18 25 33
4  1 30  2 15  2 45  1 60
0
```

Output for the Sample Input

```
Data set 1
  Order: 15 45 30 45
  Error: 0
Data set 2
  Order: 15 13 33 25 18 10
  Error: 19
```

Problem F
Typesetting

Proportional fonts are so called because characters require varying amounts of space on the printed line. The size in which text is “set,” usually measured in points, also affects the space required for each character. In this problem you are given a number of paragraphs of text to set. Each paragraph may include special “words” to select the font and point size.

The input starts with the font width table. These data give the widths of 10-point characters in six different fonts. The first line contains the number of characters in the table, N ($0 \bullet N \bullet 100$). Each of the next N lines contain a character in column 1 and then 6 integers representing the width of that character in each of the 6 different fonts.

Widths are given in an arbitrary measurement called “units.” The width of each 10-point character will be greater than zero units, and less than 256 units. Character widths scale linearly with point size. Thus if a 10-point “A” is 12 units wide, a 20-point “A” is 24 units wide.

The remainder of the input consists of paragraphs to be typeset. Each paragraph begins with a line containing two integers, *L* and *W*. *L* is the number of input lines of text for the paragraph (these immediately follow the first line), and *W* is the width allowed for each typeset line, in units. The initial font at the beginning of each paragraph is always font 1, and the initial point size in which characters are to be set is 10. Fonts are numbered 1 through 6, corresponding to columns 1 through 6 in the font width table. An empty paragraph (one for which *L* is 0) will mark the end of the input data. No output is to be produced for this empty paragraph.

The words in each paragraph are sequences of no more than 8 non-blank characters separated by spaces (that is, blanks—no tab characters will appear in the input). Spaces at the ends of input lines are irrelevant, and spaces between words are significant only to the extent that they separate words. Each character in each word will appear in the width table. Case is significant for all characters in the input data.

The special tokens "`*f1`", "`*f2`", "`*f3`", "`*f4`", "`*f5`", and "`*f6`" are used to select a particular font to be used in setting the text that follows it. The token "`*s`*N*", where *N* is an integer in the range 1 to 99 indicates that *N* point characters are to be used in setting the following text. These tokens will always be separated from words and other tokens by at least one blank. Note that style and size changes made in one paragraph do not carry over to the next paragraph, and that many such changes may appear in a single paragraph.

For each paragraph, try to set as many words per line as possible, ensuring that each word is followed by at least the width of a blank (which will always appear in the font width table) with the same point size and style as the characters in the preceding word, except for the last word on the line. The last word in a typeset line must not have any following space.

When scaling fonts, round the scaled character widths to the nearest integer, rounding upward in cases where the rounded value is half way between two consecutive integers. Thus, if a particular 10 point character occupies 9 units of space, a 15 point character would occupy 14 units of space, as would a 16 point character. A 14 point character, however, would occupy only 13 units of space.

For each paragraph, first display the paragraph number (1, 2, ...). Then, for each typeset line in the paragraph, display the line number, the first and last words on that line, and the total number of units of white space that follow the last character printed on the line. (This is just the number of units of space available on the line not occupied by characters or spaces between characters.)

If a single word exceeds the width of a line, set it on a line by itself. In the output for that line, show only that single word, and a negative amount of white space equal to the excess width of the word.

Sample Input

```
4
A 10 20 30 12 22 32
B 1 2 3 4 5 6
C 9 10 8 3 5 2
  2 4 6 3 5 7
2 80
*f2  AAA BBB CCC
  ABC *s15 CBA AABC CACA
3 100
AAA
AAA BBB CCC
ABC CBA AABC CACA
0 0
```

Output for the Sample Input

```
Paragraph 1
  Line 1: AAA ... BBB (10 whitespace)
  Line 2: CCC ... ABC (14 whitespace)
  Line 3: CBA ... CBA (32 whitespace)
  Line 4: AABC ... AABC (2 whitespace)
  Line 5: CACA (-10 whitespace)
Paragraph 2
  Line 1: AAA ... CCC (4 whitespace)
  Line 2: ABC ... AABC (26 whitespace)
  Line 3: CACA ... CACA (62 whitespace)
```

Problem G

VTAS - Vessel Traffic Advisory Service

In order to promote safety and efficient use of port facilities, the Association of Coastal Merchants (ACM) has developed a concept for a Vessel Traffic Advisory Service (VTAS) that will provide traffic advisories for vessels transiting participating ports.

The concept is built on a computer program that maintains information about the traffic patterns and reported movements of vessels within the port over multiple days. For each port, the traffic lanes are defined between waypoints. The traffic lanes have been designated as directional to provide traffic separation and flow controls. Each port is represented by a square matrix containing the distances (in nautical miles) along each valid traffic lane. The distances are defined from each row waypoint to each column waypoint. A distance of 0 indicates that no valid traffic lane exists between the two waypoints.

Vessel traffic enters the port at a waypoint and transits the traffic lanes. A vessel may begin its transit at any of the waypoints and must follow a valid connected route via the valid traffic lanes. A vessel may end its transit at any valid waypoint.

The service provided by the VTAS to transiting vessels includes:

- Projection of arrival times at waypoints
- Notification of invalid routes
- Projected encounters with other vessels on each leg of the transit. An “encounter” occurs when two vessels are between common waypoints (either traffic lane) at a common time
- Warning of close passing with another vessel in the vicinity of a waypoint (within 3 minutes of projected waypoint arrival)

Reported times will be rounded to the nearest whole minute. Time is maintained based on a 24 hour clock (i.e. 9 am is 0900, 9 PM is 2100, midnight is 0000). Speed is measured in knots which is equal to 1 nautical mile per hour.

Input

The input file for the computer program include a Port Specification to provide the description of the traffic patterns within the port and a Traffic List which contains the sequence of vessels entering the port and their intended tracks. The end of the input is indicated by a Vessel Name beginning with an "*"

Port Specification :	Number of Waypoints in Port (an integer *N*)
	Waypoint ID List (*N* single-character designators)
	Waypoint Connection Matrix (*N* rows of *N* real values specifying the distances between waypoints in nautical miles)
Traffic List:	Vessel Name (alphabetic characters)
	Time at first waypoint (on 24-hour clock) & Planned Transit Speed (in knots)
	Planned Route (ordered list of waypoints)

Output

The output shall provide for each vessel as it enters the port a listing indicating the arrival of the vessel and its planned speed followed by a table containing the waypoints in its route and projected arrival at each waypoint. Following this table will be appropriate messages indicating:

- Notification of Invalid Routes
- Projected Encounters on each leg
- Warning of close passing at waypoints

All times are to be printed as four-digit integers with leading zeros when necessary.

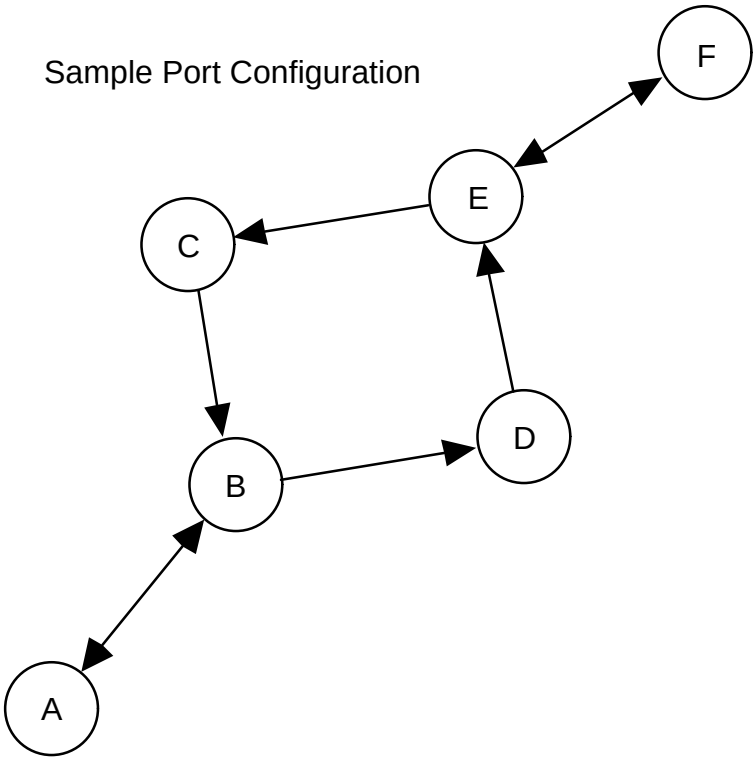
Assumptions & Limitations

1. *Vessel names are at most 20 characters long.*
2. *There are at most 20 waypoints in a port and at most 20 waypoints in any route.*
3. *There will be at most 20 vessels in port at any time.*
4. *A vessel will complete its transit in at most 12 hours.*
5. *No more than 24 hours will elapse between vessel entries.*

Sample Input

```
6
ABCDEF
0 3 0 0 0 0
3 0 0 2 0 0
0 3 0 0 0 0
0 0 0 0 3 0
0 0 2 0 0 4
0 0 0 0 4 0
Tug
2330 12
ABDEF
WhiteSailboat
2345 6
ECBDE
TugWBarge
2355 5
DECBA
PowerCruiser
   0 15
FECBA
LiberianFreighter
   7 18
ABDXF
ChineseJunk
  45 8
ACEF
*****
```

Sample Port Configuration

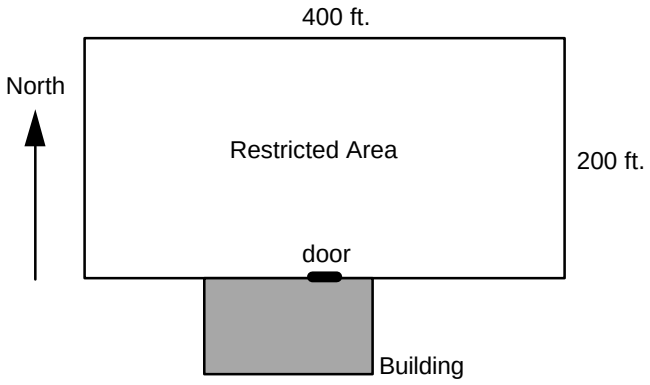


Output for the Sample Input

```
Tug entering system at 2330 with a planned speed of 12.0 knots
          Waypoint:    A    B    D    E    F
          Arrival:   2330 2345 2355 0010 0030

WhiteSailboat entering system at 2345 with a planned speed of 6.0 knots
          Waypoint:    E    C    B    D    E
          Arrival:   2345 0005 0035 0055 0125

TugWBarge entering system at 2355 with a planned speed of 5.0 knots
          Waypoint:    D    E    C    B    A
          Arrival:   2355 0031 0055 0131 0207
Projected encounter with Tug on leg between Waypoints D & E
** Warning ** Close passing with Tug at Waypoint D

PowerCruiser entering system at 0000 with a planned speed of 15.0 knots
          Waypoint:    F    E    C    B    A
          Arrival:   0000 0016 0024 0036 0048
Projected encounter with Tug on leg between Waypoints F & E
Projected encounter with WhiteSailboat on leg between Waypoints C & B
** Warning ** Close passing with WhiteSailboat at Waypoint B

LiberianFreighter entering system at 0007 with a planned speed of 18.0 knots
**> Invalid Route Plan for Vessel: LiberianFreighter

ChineseJunk entering system at 0045 with a planned speed of 8.0 knots
**> Invalid Route Plan for Vessel: ChineseJunk
```

Problem H
Monitoring Wheelchair Patients

A researcher at a rehabilitation facility is studying the use that a patient makes of a motorized wheelchair in a restricted area at the facility. The chair's motor is connected to the axle by a chain drive. Therefore both wheels turn at the same speed and the chair can travel only in a straight line. The patient can stop the chair, rotate the wheels, and thereby change the direction only while the wheelchair is stopped. To help monitor its usage, the chair is equipped with a compass, a clock, and a speedometer. A recording device records each time interval that the chair is in motion, the average speed during the time interval, and the compass bearing during the time interval. The compass is a standard compass in which 0• is north, 90• is east, and so forth.



A map of the restricted area is shown. The restricted area is a 200 ft by 400 ft rectangular area of the lawn. Patients enter the restricted area from the door of a building located on the southern edge of the restricted area. The door is at the center of the 400 ft southern boundary, as shown in the figure.

The recording device turns itself on when the patient enters the restricted area through the door and monitors the patient's movements for up to 1 hour. Time is measured in seconds from 0 to 3600, with time 0 being the time the patient initially enters the restricted area through the door. The device records 4 numbers to describe the motion of the wheelchair during any interval when the motor is in operation. The first two numbers give the time the motion begins and ends; the third number gives the speed during the time interval; and the fourth number gives the compass bearing during the time interval. (During each time interval the wheelchair maintains constant speed and bearing.) For example, the recorded line

```
10.6  15.9  2.8   274
```

would indicate that between times $t_1 = 10.6$ and $t_2 = 15.9$ seconds the wheelchair was traveling at speed of 2.8 ft/sec with compass bearing (direction) 274•. Times are recorded to 0.1 sec, speeds are recorded to 0.1 ft/sec, and bearings are recorded to a whole number of degrees.

Your job is to analyze the data from the wheelchair's recording device. Specifically, you must determine the following:

1) Did the patient ever leave the restricted area? If so, determine the first time that the patient left the restricted area and determine at what point on the perimeter of the restricted area the wheelchair crossed out of the restricted area. If the patient did not leave the restricted area, what was the distance from the door to the farthest point the patient reached within the area?
2) What was the total distance that the patient traveled?

For the purpose of answering these questions, use coordinates with the location (0,0) corresponding to the southwest corner of the restricted area and the location (400,200) corresponding to the northeast corner. Since the recorder switches on when the patient passes through the door, the position of the patient at time t = 0.0 is always (200,0). Patients will be traveling north when they enter the restricted area.

Input

The input data consists of several data sets. The first line of each data set has an integer which is the number of lines recorded by the device. Each subsequent line in the data set consists of the four numbers recorded by the device during a particular time interval. The end of data is indicated by a data set whose first line consists of the number 0.

In the first data set of the sample input, the patient entered through the door (at time 0.0) and for the first 5 seconds was traveling due north at 3 ft/sec. From time t = 7 to t = 9 he traveled at a speed of 2 ft/sec with a compass bearing of 30•. He then stopped, changed his bearing to 60•, and then traveled at 4 ft/sec from time t = 10 to time t = 100. Ten seconds later (at time t = 110) he headed due north at 2 ft/sec until t = 200.

Output

The output for each data set begins with an identification of that case. The output indicates whether the patient departed from the restricted area and if so the time and point of departure on the perimeter . If not, the maximum distance the patient reached from the door is provided. For each case, the total distance that the patient traveled is provided. Format your output so that the same labeling information is included as shown in the sample output, with a line of asterisks separating the cases.

Sample Input

```
4
0.0 5.0      3.0   0
7.0 9.0      2.0   30
10.0  100.0 4.0   60
110.0 200.0 2.0   0
3
0.0       20.0      2.0       0
500.0     600.0     1.0       270
3000.0    3100.0    1.0       0
7
0.0       5.3       2.1       0
19.8      35.6      2.7       346
42.0      78.4      2.3       15
1181.4    1192.1    1.7       117
2107.0    2193.6    2.1       295
2196.3    2201.2    2.0       298
2704.3    2709.2    1.5       208
0
```

Output for the Sample Input

```
Case Number 1
Left restricted area at point (400.0,132.8) and time 67.2 sec.
Total distance traveled was 559.0 feet
**************************************
Case Number 2
No departure from restricted area
Maximum distance patient traveled from door was 172.0 feet
Total distance traveled was 240.0 feet
**************************************
Case Number 3
Left restricted area at point (67.0,200.0) and time 2191.4 sec.
Total distance traveled was 354.7 feet
**************************************
```

Assumptions and requirements

1. *Within each data set, time intervals will be listed in chronological order, with the first time interval always having time 0.0 as the time of entry into the restricted area. All times will be given with one decimal place accuracy and will be in the range 0.0 to 3600.0 inclusive. For each time interval specified, the duration of the time interval will be positive, i.e. the second time specified will be greater than the first.*
2. *Speeds will be in the range 0.1 to 9.9 ft/sec.*
3. *Compass bearings will be given as a whole number of degrees and will be in the range 0 to 359 inclusive. The initial compass bearing for the first line of data in each data set will be 0.*
4. *Within each line of data, numbers will be separated by at least one blank space.*
5. *All numerical results will be displayed with one decimal place of accuracy as shown in the sample output.*
6. *If the patient goes out of the restricted area, his location may include negative coordinates. However, you don't have to worry about the wheelchair crashing through the walls of the building.*