1986

ACM SCHOLASTIC PROGRAMMING CONTEST

AIRLINE HUBS

Program File:	**HUBS.PAS**	--> Pascal Version
	HUBS.FOR	--> FORTRAN Version
Input Data File:	**HUBS.DAT**	
Output File:	**HUBS.OUT**	

In recent years the major airlines have adopted a "hub and spoke" route structure to provide more economical schedules. A central city is chosen as the hub; morning flights feed passengers from all other cities into the hub, and afternoon flights take the passengers from the hub to all other cities. This problem asks you to help Bogus Airlines choose their hub.

Bogus Airlines (BA) expects to serve up to 30 cities. It knows the following information about flight scheduling. The flying time between two cities is the distance divided by 500 mph, plus 30 minutes. Every flight must be allowed 30 minutes ground time before takeoff and 30 minutes ground time after landing. Thus, for example, a 1000 mile flight must be scheduled to take three and a half hours (1000/500 + 0.5 + 0.5 + 0.5 hours). There must be at least one hour at the hub between connecting flights. However, it is not necessary to have one hour between a morning flight from a particular city and an afternoon flight back to that same city; we assume no passenger would take both flights. No flight may begin before 8:00 am, nor terminate after 10:00 pm. The selection of a city as a hub will always allow any passenger to travel from any one city to the hub and then on to any other city in one day (i.e., between 8 am and 10 pm). You may also assume that an airport can accommodate any number of simultaneous arrivals and departures.

For purposes of this problem, you may interpret the words "morning" and "afternoon" liberally. For example, an "afternoon" flight might actually be scheduled before noon or after 6:00 pm as long as the other flight conditions presented in the previous paragraph are satisfied.

You will be given five data sets in a single file. For simplicity, the locations will be given as coordinates in a plane (no spherical geometry will be needed). Latitude and longitude will each be expressed as an integer in the range 0 to 9999, and the unit of measurement is a mile. Distances between cities may be computed with the normal Euclidean distance function. The first line of each data set contains a two digit number in the first two character positions of the line; that is the number of cities. The rest of the data set consists of lines describing one city each. The first eight characters on the line are the name of the city (uppercase alphabetic characters, digits, and blanks only, left justified). Immediately after the name are two five-character fields, holding latitude and longitude, respectively, of the city; these values are right-justified integers. Each subsequent data set will follow immediately after its predecessor.

For each data set, your program must identify every city that could be a hub, meaning that it would allow scheduling of flights as described above. Then your program must identify the city that would be the best hub. Such a city is one that would allow the shortest total time from the first flight of the morning (at 8:00 am) to the end of the last flight of the afternoon (including, of course, ground times as described above). If two or more cities are equally good, identify all of them. Your program must also print the number of possible hubs and number of best hubs for each data set. The sample input and corresponding output should help you understand what you are to do.

Sample Input

```
06
CITY 1     200   400
CITY 2     100   300
CITY 3     100   200
CITY 4     200   100
CITY 5     300   200
CITY 6     300   300
```

Corresponding Output

```
6 POSSIBLE HUB(S) MAY BE AT
CITY 1
CITY 2
CITY 3
CITY 4
CITY 5
CITY 6

4 BEST HUB(S) AT
CITY 2
CITY 3
CITY 5
CITY 6
```

1986

ACM SCHOLASTIC PROGRAMMING CONTEST

DATABASE QUERIES

Program File:	**QUERIES.PAS**	--> Pascal Version
	QUERIES.FOR	--> FORTRAN Version
Input Data File:	**QUERIES.DAT**	
Output File:	**QUERIES.OUT**	

In many environments, it is desirable for naive users to be able to access information in a database through natural language queries. For this problem you will implement both a small database and a query language interpreter.

The database will describe who and what object are in each room of a house. Each person, object, and room has a name that is a single word, all uppercase, of at most ten characters. There are exactly eight rooms in the house, and every room contains exactly one person and one object. The database will be described in the first eight lines of data. Each of these lines contains a room name left-justified in the first ten positions of the line, a person's name left-justified in the next ten positions, and an object's name left-justified in the next ten positions. For example:

```
LIVINGROOMSMITH     PIANO
BATHROOM  JONES     TOOTHBRUSH
```

The query language permits just four types of questions:

```
WHO IS IN THE <room>?
WHAT IS IN THE <room>?
WHERE IS <person>?
WHERE IS THE <object>?
```

Questions will be expressed in all uppercase letters, one question per line. Your program must read a question, print one blank line, print the question on the next line, and then on the following line print either the one-word answer or an error message. If the question is phrased incorrectly, then print the error message ERRONEOUS QUESTION. If a correctly phrased question mentions a room, person, or object not in the database, print the error message UNKNOWN.

Answers appear on a separate line immediately after the question. Questions will begin in the ninth input line, after the eight database lines. Questions will have at least one space between words, and may have spaces before the first word or before the question mark. Question words will contain no more than ten characters. The question mark must be present; otherwise, the question is erroneous. Characters following the question mark should be ignored and are not to be printed with the question. There will be an arbitrary number of question lines, terminated by end-of-file.

For example, with a database containing values from the two lines above:

```
WHERE IS JONES?
BATHROOM

   WHAT   IS   IN   THE   LIVINGROOM   ?
PIANO

WHAT IS IN THE VARDAGSRUM?
UNKNOWN

WHO ATE MY WOMBAT?
ERRONEOUS QUESTION
```

1986

ACM SCHOLASTIC PROGRAMMING CONTEST

GETTING THE MESSAGE

Program File:	**MESSAGES.PAS**	--> Pascal Version
	MESSAGES.FOR	--> FORTRAN Version
Input Data File:	**MESSAGES.DAT**	
Output File:	**MESSAGES.OUT**	

The Midnight Message Service records messages for its clients each night in a single text file. The first line of the file contains (in the first twenty-five character positions) the date on which the file of messages were received. The remainder of the file contains the messages in the order they were received. Each message is comprised of two lines. The first line contains the account number of the client to whom the message was sent. The second line contains the message to that client. No message is longer than 72 characters and the account number is a positive integer less than 32767. There are at most 200 messages in the file.

Write a program that produces a report containing all messages in the file. The report will consist of a heading followed by a sequence of client reports appearing in ascending order by account number. The heading and the first client report will be separated by two blank lines; client reports will be separated by one blank line. The heading will consist of the line below with <date> replaced by the date the messages were received.

Midnight Message Service -- messages received on <date>.

Client reports will be single spaced. The first line of each client report will contain the client's account number. The remaining lines will list the client's messages, one per line, in the order received.

1986

ACM SCHOLASTIC PROGRAMMING CONTEST

Lobid Code

Program File:	**LOBID.PAS**	--> Pascal Version
	LOBID.FOR	--> FORTRAN Version
Input Data File:	**LOBID.DAT**	
Output File:	**LOBID.OUT**	

Your employer, Lobid Inc., wishes to propose an encryption and decryption scheme by which a message text may be encrypted as a sequence of integers. The text is first partitioned into 15-character blocks, each of which is then encrypted as a sequence of five integers. (If the length of the message is not a multiple of fifteen, it is padded at the end with blanks.) Only the 32 characters appearing in the following example may be used in a message.

The message 'ABCDEFGHIJKLMNOPQRSTUVWXYZ .,()-' is encrypted as

0, 255, 3855, 13107, -10923, -1, 127, 1927, 6553, 10922, -8193, -8193, -16384, -8193, -8193.

The encryption procedure is illustrated below:

block 1 --> A B C D E F G H I J K L M N O

bit 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	yields 0
bit 2	0 0 0 0 0 0 0 1 1 1 1 1 1 1 1	yields 255
bit 3	0 0 0 1 1 1 1 0 0 0 0 1 1 1 1	yields 3855
bit 4	0 1 1 0 0 1 1 0 0 1 1 0 0 1 1	yields 13107
bit 5	1 0 1 0 1 0 1 0 1 0 1 0 1 0 1	yields -10923

block 2 --> P Q R S T U V W X Y Z . , (

bit 1	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	yields -1
bit 2	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1	yields 127
bit 3	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1	yields 1927
bit 4	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1	yields 6553
bit 5	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0	yields 10922

block 3 -->) -

bit 1	1 0 1 1 1 1 1 1 1 1 1 1 1 1 1	yields -8193
bit 2	1 0 1 1 1 1 1 1 1 1 1 1 1 1 1	yields -8193
bit 3	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	yields -16384
bit 4	1 0 1 1 1 1 1 1 1 1 1 1 1 1 1	yields -8193
bit 5	1 0 1 1 1 1 1 1 1 1 1 1 1 1 1	yields -8193

Note from the example that each character is represented by a unique 5-bit code. Furthermore, the first of the five integers representing a block is derived by interpreting the first bits of each of the block's characters as a 15-bit 2's-complement integer. The 15 second bits are combined in a like manner to form the second integer, and so forth.

The data file for your program is in two parts. The first part is a sequence of one or more lines, each containing 16 characters in the first 16 columns. For each line, write the sequence of five integers generated by encrypting the first 15 characters. Write these code numbers in the proper order on a single output line. If the 16th character is a plus sign ("+"), another line to be encrypted follows; otherwise, the encryption phase is over.

The remainder of the file consists of zero or more lines, each containing five integers derived from encoding a block of message text. These integers are right-justified in five consecutive fields of seven characters each. You may assume there is at least one blank between adjacent integers. For each of these lines, write the decoded block of text on a single line.

1986

ACM SCHOLASTIC PROGRAMMING CONTEST

PICTURE MAKER

Program File:	**PICTURE.PAS**	--> Pascal Version
	PICTURE.FOR	--> FORTRAN Version
Input Data File:	**PICTURE.DAT**	
Output File:	**PICTURE.OUT**	

An upstart new company, Pseudo-Art, Inc. has hired you as a hotshot programmer to develop a program that will generate pictures containing an embedded message on an ASCII printer. The program will take a message and a picture specification as input and create a composite picture for a customer.

The generated pictures are to be constructed from the characters contained in the customer's message and displayed in a 24 line by 64 column matrix format. The message may be up to 32 characters in length with no blanks and is to be inserted into the matrix repeatedly row by row until the entire area is filled. The picture specification consists of a list of numbers indicating the number of positions counted row by row to be alternately masked by blanks or printed as message characters within the matrix. The first number in the file indicates that the field is to be printed as message characters.

INPUT SPECIFICATION:

A single data set is contained in a file arranged as follows:

Record #	Contents
1	Message to be embedded in the picture (up to 32 characters long)
2-n	Integer numbers, 10 per line right justified in 5 character fields indicating the field size to be printed alternately as message characters or blanks. The last value is a 0 indicating the end of data.

OUTPUT SPECIFICATION:

Your program shall create a picture in the matrix composed of the message characters and blanks specified by the input file and write the resultant matrix to a file. The characters in the matrix are to be written with 24 lines of 64 columns each to the file.

EXAMPLE:

A sample data set:

```
BILBO.BAGGINS.FOR.PRESIDENT
    4   56    8   56    4   76   35   29   35   29
   35   45    6   58    6   58    6   58    6   58
    6   58    6   58    6   52   18   46   18   46
   18  244    6   58    6   30    4   24    6   26
    8   56    8   56    4    0
```

Output generated by above data:

```
BILB                                                          BAGG
INS.                                                          R.PR

          OR.PRESIDENTBILBO.BAGGINS.FOR.PRESI
          NTBILBO.BAGGINS.FOR.PRESIDENTBILBO.
          GGINS.FOR.PRESIDENTBILBO.BAGGINS.FO
                          AGGINS
                          BILBO.
                          INS.FO
                          ESIDEN
                          BO.BAG
                    BILBO.BAGGINS.FOR.
                    INS.FOR.PRESIDENTB


                          LBO.BA
                          S.FOR.
SIDE                      IDENTB                              BILB
O.BA                                                          INS.
FOR.                                                          ESID
```

1986

ACM SCHOLASTIC PROGRAMMING CONTEST

PROGRAMMING STYLE METRICS

Program File:	**STYLE.PAS**	--> Pascal Version
	STYLE.FOR	--> FORTRAN Version
Input Data File:	**STYLE.DAT**	
Output File:	**STYLE.OUT**	

In an attempt to help automate the process of software quality assurance, researchers are atttempting to find metrics or measures of what is good or bad about programs. Several metrics of programming style have been proposed. This problem involves some of the simplest of those metrics.

Your program should read as data the source code of a Pascal program. It should count the following things:

1. The total number of lines in the program.

2. The number of blank lines in the program.

3. The total number of characters in the program.

4. The number of characters in comments, including the comment delimiter characters.

5. The number of blank or space characters in the program, excluding blanks in comments or in character string literals.

6. The number of other characters in the program (not blank, not in comments).

Your program should display each of these counts, appropriately labeled.

The following information may be useful to refresh your memory about the structure of Pascal programs. A program is free format and lines can be arbitrarily short or long. A comment begins with a { character, and terminates with a } character. All characters between the two delimiters are part of the comment, as are the delimiters. For the purposes of this program, no nesting of comments will occur. A string literal begins and ends with an apostrophe (') character. Two consecutive appostrophes in the literal are interpreted as a single apostrophe, rather than the terminating delimiter. For counting purposes, however, two consecutive apostrophes count as two characters. Braces ({ or }) inside a string literal are not comment delimiters, nor is an apostrophe inside a comment the delimiter of a string literal.

The data set for this problem is a correct Pascal program. It may be arbitrarily long, and is terminated by end of file. No line in the data set has trailing blanks. If you choose to solve this problem in a way that introduces trailing blanks, you must also remove them. No line in the data set exceeds 80 characters. Control characters such as carriage return, line feed, or newline are not to be counted in any way. There are no tab characters in the data set. No correct counts will exceed the maximum integer value.

Sample Input:

```
program test ( input, output );

  { this is a data set
  }
var
    num  : integer;
    word : packed array [ 1..15 ] of char;
begin
  word := ',:.            ';  { this is a comment }
  for num := 1 to 5 do
      writeln ( 'Number is:', num * 4 : 3 );
  word := '{not a comment}';
end.  { end test }
```

Corresponding Output:

```
Total lines              13
Blank lines               1
Total characters        288
Blank characters         60
Comment characters       56
Other characters        172
```

1986

ACM SCHOLASTIC PROGRAMMING CONTEST

RAILROAD SCHEDULING

Program File:	**RAILROAD.PAS**	--> Pascal Version
	RAILROAD.FOR	--> FORTRAN Version
Input Data File:	**RAILROAD.DAT**	
Output File:	**RAILROAD.OUT**	

The Binary and Octal (B&O) Railroad has decided that they are going to have to become more efficient on their freight routes if they are going to be able to compete in the current tight marketplace. B&O is a small railroad company with a central station and 10 outlying stations along their lines. Each of the stations is directly connected to each other and B&O knows of the scheduled cargo to be shipped to each station at least one day in advance.

Each day the B&O Bit Bending Special starts from the central station with the scheduled load of cars to be delivered to their scheduled destinations. On any given day there may be up to 100 cars delivered to each destination from the central station and a delivery may go to each of the stations. No cars are delivered to the central station.

The problem that the B&O management has asked for your help in solving is to schedule the daily deliveries to provide the minimum cost routing for the daily train. The minimum cost is defined to be the summation of miles traveled times number of cars carried. Since the number of cars for each destination and number of destinations varies daily, it will be necessary to recalculate the minimum cost route daily.

INPUT SPECIFICATION:

A single data set is contained in a file arranged as follows:

Record #	Contents
1	Number of Destinations (N) for today's deliveries
2	1 to N integers right justified in 5 character fields, representing the number of cars to be delivered to each destination.
3 : : 3 + N	Integer numbers right justified in 5 character fields, representing the distances between stations for today's run. The distances are presented as an upper triangular matrix containing only necessary distances. For example, as the first record with N = 3 the values 1 2 3 appear. These values represent the distances from stations 0 to 1, 0 to 2 and 0 to 3 (station 0 is the central station and the receiving stations are stations 1-N).

OUTPUT SPECIFICATION:

Your program is to determine the minimum cost route for the B&O daily train and write the minimum cost and route to be taken to a file. The exact format for your output is shown in the sample output.

EXAMPLE:

A sample data set:

```
4
   70    50    16    21
    4     5     6     1
    4     4     5
    1     2
    3
```

Output generated by above data:

```
  Minimum cost is 795 car/miles.

  Stop 1 is Station 4
  Stop 2 is Station 2
  Stop 3 is Station 3
  Stop 4 is Station 1
```

1986

ACM SCHOLASTIC PROGRAMMING CONTEST

RSA CYPHER DECRYPTION

Program File:	**RSA.PAS**	--> Pascal Version
	RSA.FOR	--> FORTRAN Version
Input Data File:	**RSA.DAT**	
Output File:	**RSA.OUT**	

Recently there has been renewed concern about the security of information being processed by computers and the transmission of that information between computer sites. When confidentiality is needed, information is typically encoded in some way with a cypher, rendering it unintelligible to all but the intended user. The problem with most cypher schemes is that a 'key' to decode the information is known to both sender and receiver. This raises the issue of how to transmit the 'key' in a secure manner. One system that avoids this difficulty is the RSA public key encryption system, which is based on the use of a 'trap door' mathematical function, and the difficulty associated with the factorization of large numbers into their prime factors.

The RSA system is a two-key system. The public key is known by all users. Each individual user is then given a private key. To send a message, the message is encoded using the public key of the intended receiver and transmitted to him. He then applies the decryption process to the message, using both his public and private keys. The essential feature of the RSA system is that decrypting a message using the same key as was used to encrypt it does not yield the original message. The private key must also be used in the decryption process.

You are to develop a program to implement a limited version of the RSA decryption process. The keys which will be used are too small for reasonable data integrity, but the method is valid.

The input data file is a text file consisting of a list of sets of data, one set for each line of the original message. Each data set is a sequence of unsigned integer values, arranged one per line, and right-justified in the first five columns of successive lines of input. The first value of all but the last data set contains the number of values in the data set, including itself. The last data set consists of only one line, containing the value 0. For each data set, the numbers after the first are encrypted blocks to be decoded as described below.

Decrypting an encrypted block is a two step-process. First, the encrypted block, EB, is decoded into a decimally-encoded block, DEB, using the transformation,

DEB = (EB ** DK) mod EK ,

where DK is the private decryption key, EK is the public encryption key, and '**' denotes exponentiation. The non-negative decimal number DEB is then decoded into two characters of the original message. The order (ASCII character code) of the first character is 31 plus the integer in the range 0 to 99 whose tens and ones digits are DEB's thousands and hundreds digits, respectively. The order of the second character is 31 plus the integer in the range 0 to 99 whose tens and ones digits are DEB's tens and ones digits. For example, using the private decryption key 26787 and the public encryption key 40753, the encrypted block 30239 yields the decimally-encoded block 7865. This is decoded into "NA", since the order of 'N' is 78 and the order of 'A' is 65.

In order to compute DEB=(EB**DK) mod EK without exceeding the precision of the machine, a modified version of the "Russian peasant" algorithm may be used:

1. Let DEB be 1.
2. If DK is odd, change DEB to (DEB * EB) mod EK.
3. Change EB to (EB * EB) mod EK.
4. Change DK to the integer part of (DK divided by two).
5. If DK is zero, stop. Otherwise, repeat from step 2.

You are to decrypt, reassemble, and print any message encoded with the keys mentioned in the example. Preserve the the line structure of the original message. Assume that no decrypted line is longer than 80 characters.

HINT: Note that DEB is always less than EK. Consider this and the arithmetic operations required in the algorithm given above when selecting your data representation.