1984 ACM INTERNATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem : Bridge Point Counting

A "hand" in bridge consists of 13 cards. Bidding begins by adding the point value of the hand and then bidding if it is appropriate.

You are to write a program which will input the 13 cards in a bridge hand and then calculate the point value of the hand. Each card is represented by two two-digit numbers. Each number is separated from the preceding number by one blank. The first number specifies the rank of the card: 1=ace, 2=deuce, ..., 10=ten, 11=jack, 12=queen, 13=king. The second number specifies the suit: 1=clubs, 2=diamonds, 3=hearts, 4=spades. Thus:

```
 9  3  =  9 of hearts
 1  2  =  ace of diamonds
13  4  =  king of spades
```

The values representing the thirteen cards of a hand will all be on the same input line, alternating rank, suit, ..., rank, suit.

Calculate the point value of a hand by using the following rules:

a. 4 points for an ace
b. 3 points for a king in a suit with at least one other card (i.e., at least two cards in the suit)
c. 2 points for a queen in a suit with at least two other cards (i.e., at least three cards in the suit)
d. 1 point for a jack in a suit with at least three other cards (i.e., at least four cards in the suit)
e. 3 points for a suit with no cards
f. 2 points for a suit with only one card
g. 1 point for a suit with only two cards
h. apply rules f. and g. even if one of the other rules applies
i. apply rules a. - d. as often as appropriate in any suit

Example: If the hand has K Q J of hearts
  add 3 points for the K
  add 2 points for the Q
  no points for the J, since there are only 3 cards in the suit

Example: If the hand has K 3 of spades
  add 3 points for the K
  add 1 point for a suit with only 2 cards

For each hand, input the card values, calculate the point value, and print the hand and the point value for that hand with appropriate identification. The program should terminate after all hands have been processed.

1984 ACM INTERNATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 6: Optimum Prefix Codes

For a given set of values, define a "code" as a set of symbols to represent those values. A "symbol" is defined as a string of digits chosen from the set 0, 1, ..., k-1. Not all symbols will necessarily contain the same number of digits.

An "optimum prefix code" is a code with two properties:
(1) no symbol is the prefix of any other symbol, and
(2) all symbols are as short as possible

For example, a set of ten values may be coded with symbols chosen from 0, 1 as:

0000 0001 010 011 100 101 110 111 0010 0011

This code has the two properties, and is therefore an optimum prefix code.

The following example shows another way to code ten values with symbols made up of 0, 1, but it does not have property (1) above.

0 1 00 01 10 11 000 001 010 011

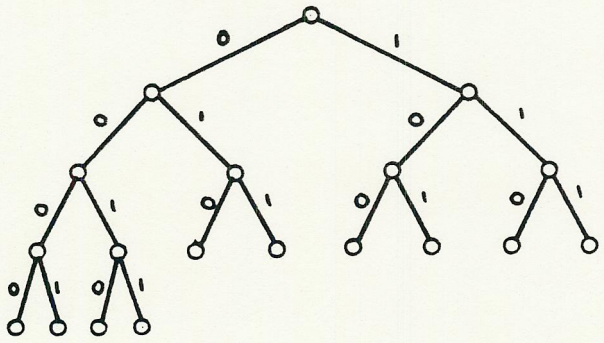
The following example also shows a way to code ten values with symbols made up of 0, 1, but it does not have property (2) above.

0 10 110 1110 11110 111110 1111110
11111110 111111110 1111111110

Your are to write a program that will find and print optimum prefix codes. The input will consist of n, the number of values to be represented, and k, the number of digits allowed in symbols. These values will be integers with $2 \leq n \leq 100$ and $2 \leq k \leq 8$. The value of n will be right justified in columns 1 to 3, and the value of k will be right justified in columns 4 to 6, both on the same line. The above example has n = 10, k = 2. There will be exactly four data sets.

For each data set, print the values of n and k, appropriately labeled, and then the code symbols, with eight symbols per output line (except perhaps the last line). The symbols may be printed in any order.

Notice that if the symbols of an optimum prefix code are arranged in a tree, with each level in the tree adding one more digit to a symbol, then an optimum prefix code has the minimum number of levels, and every symbol corresponds to a leaf node of the tree. The first code example above is shown in the diagram.



1984 ACM INTERNATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 5: Honor Society Selection

Imatellin University has been granted a chapter of Upsilon Gamma Eta, the national honor society for students in All-Night Bowling Alley Management. Imatellin U. needs a program to select those students who are eligible to join this august society. The criteria are:

(1) 30 or more credit hours of courses attempted and a 3.0 or higher grade point average in those courses, and

(2) 12 or more credit hours of Bowling Alley Management courses attempted and a 3.25 or higher grade point average in those courses.

The course history file for Imatellin U. students contains for each student a contiguous group of 68-character records, each containing the student's name in the first 20 characters immediately followed by up to eight adjacent course entries. All alphabetic characters are upper case.

Each 6-character course entry consists of a two-letter department prefix (BM for Bowling Alley Management courses), a two-digit course number, one digit specifying the number of credit hours the course carries, and the letter grade received. Unused course entries are all blanks.

Grade point averages are to be computed on a four-point basis (A=4, B=3, C=2, D=1, F=0), weighted as usual by the number of credit hours for each course. To keep its cherished accreditation, Imatellin does not have any other grading basis such as Pass/Fail, and every attempt at every course is included in the grade point calculation.

Write a program to read a data file as described above, and list the names of those students in the file who qualify for Upsilon Gamma Eta.

An example student record group is given below.

```
KEGLER, STAR          PE014AHI013DMA024CBM014ASS223F
KEGLER, STAR          SS223CPE024AMA044DBM024AHI023DTX031D
```

1984 ACM INTERNATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 4: Expanding Square Codes

One method of encoding messages is known as the "expanding square code". This method encodes messages by placing the characters of the message in an odd order square matrix row by row, and then retrieving them in a clockwise expanding square spiral from the center of the matrix. If the message is not exactly the right length to fill the matrix, the rest of the matrix is filled with asterisk characters (*).

For example, the two square matrices below show the order in which the characters are placed in the matrix, and the order in which they are retrieved. Notice that the order of retrieval begins at the center, then proceeds to the right, and then spirals clockwise.

Order In					Order Out				
1	2	3	4	5	21	22	23	24	25
6	7	8	9	10	20	7	8	9	10
11	12	13	14	15	19	6	1	2	11
16	17	18	19	20	18	5	4	3	12
21	22	23	24	25	17	16	15	14	13

Message In: "This is a test message!!"

Message Out: "stssees a a*!!egmtiThis"

Your program must be able both to encode and to decode messages by this method. The input will consist of pairs of lines. The first line in each pair will contain either the word ENCODE or the word DECODE in upper case characters beginning in column 1. The second line in each pair will consist of a message to be encoded or decoded, containing a maximum of 72 characters.

From the length of the message, you should determine the minimum odd order required for the matrix, and perform the specified operation. The output for each operation will consist of one blank line, the given message, and the resultant message. Each message should be on a separate line, and each should be appropriately labeled. A decoded message may not contain the asterisk characters used to fill the matrix in the encoding process. You may assume the no actual message ends with an asterisk.

Your program should continue reading input lines and performing operations until an operation other than ENCODE or DECODE is encountered.

1984 ACM INTERNATIONAL SCHOLASTIC PROGRA'MING CONTEST

Problem 3: Bugs

The notorious scarfing bug (of the order hemiptera, scientific name Eatibus anythingus), native of northern New Jersey, exhibits an interesting behavior around others of its species. When placed on a flat surface with another bug in its sight, it proceeds to walk at a constant speed directly toward the other bug in an attempt to catch and eat it. If the second bug has a third bug in its sight, it does the same thing; similarly for the third bug. When several bugs are placed on a surface together, they each proceed along a (usually) curved path toward another moving bug.

For this problem, you will be given the starting coordinates of several bugs, and you must determine where they end up. There will be exactly four data set to process, each structured as follows:

> The first line contains an integer from 2 to 9 (inclusive) in column 1, representing the number of bugs. The next several lines (one line per bug in the order Bug1, ..., BugN) contain two real numbers each, representing the x and y coordinates, respectively, of the starting positions of the bugs. Each number will be in the range -9999.9 to +9999.9, each will contain a decimal point and one digit to the right, and they will be right justified in columns 1 through 8 and 9 through 16.

Each bug sees only the next higher numbered bug (the highest numbered bug sees only bug number 1). Simultaneously, each takes a step of length one unit directly toward the bug it sees. For purposes of this problem, this continues until either:

(1) at least one bug reaches a position less than or equal to one unit away from the bug it is stalking; or
(2) 100 steps have been taken

As soon as either of the above conditions exists, print:

> two blank lines;
> the number of steps taken;
> the bug number and the final position (x and y coordinates) of each bug, accurate to one decimal place, one bug per line

Notice that after each step, each bug may need to change direction, since its target bug is also moving. Notice also that if the starting position of a bug is less than or equal to one unit from its target, then no steps are taken.

1984 ACM INTERNATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 2: Automatic Spelling Correction

Studies have shown that the most common kinds of spelling errors made by users of text editors or word processing systems are often typing errors, specifically:

(1) transposing two adjacent letters
(2) mistyping one letter
(3) omitting one letter

With the aid of a dictionary, word processing systems can often detect and automatically correct these kinds of spelling errors. You are to write a program that does part of the job of locating spelling errors, namely recognizing the three kinds of errors described above. For this problem, a "word" will consist of 1 to 10 upper case alphabetic characters, and each input word will begin on a new line, in column 1.

The input will contain first up to 100 words; these will constitute the "dictionary". The dictionary words will be terminated by an input line with an asterisk (*) in column 1 and blanks in columns 2 to 10.

After the dictionary lines will be words whose spelling is to be checked. For each of these words, first print the word being checked, at the beginning of a new line, and then print one or more messages about it, chosen from the following:

If the word is in the dictionary correctly spelled, print CORRECT and proceed immediately to the next input word.

If the word is not in the dictionary, then find each word in the dictionary for which the given word might be a misspelling. Print as many of the following messages as are appropriate:

TWO LETTERS TRANSPOSED IN word
ONE LETTER DIFFERENT FROM word
ONE LETTER OMITTED FROM word

Note that two or more of these messages might be appropriate, and that one message might apply for more than one dictionary word. Print all messages that are valid. In each case, the "word" in the messages above is the word from the dictionary. These messages should be printed one per line.

If none of these messages is appropriate, print UNKNOWN.

All messages should be indented four spaces. Continue reading words and printing messages until you reach an end of file condition for the input file.

1984 ACM INTERNATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 1: Additive Decomposition

You are to read a series of input lines, each containing two integers; call these integers "sum" and "terms" for reference. "Sum" will be right justified in the first two columns of the input line, column three will be blank, and "terms" will be right justified in the fourth and fifth columns. For each pair of "sum" and "terms" you are to print a line containing the values of "sum" and "terms" and the number of different ways that "sum" may be represented as the sum of "terms" positive integers.

For example, six may be represented as the sum of one positive integer in only one way, namely 6; six may be represented as the sum of two positive integers in three different ways: 5+1, 4+2, and 3+3. (Notice that 2+4 is not essentially different from 4+2.) For this example, your program would produce a line of output similar to the following:

6 MAY BE REPRESENTED AS THE SUM OF 2 INTEGERS IN 3 WAYS

Your program should read pairs of integers and produce output until a pair of integers is read where "sum" = 0 and "terms" = 0; you then stop.

You may assume that the input values are within the following bounds:

$0 < \text{"sum"} < (31-\text{"terms"})$

$0 < \text{"terms"} < 11$