1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

The Battle of Waterloo

held at

The University of Waterloo
5-6 November 1993

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Some rules and advice

1. All questions require you to read the test data from standard input and write results to standard output.

   - No whitespace should appear at the end of a line, and all lines should be terminated with a new-line.
   - Tabs should never be used.
   - Output must correspond exactly to the provided sample output, including spelling and spacing. Multiple spaces will not be used in any of the judges output.

2. All programs will be re-compiled prior to testing with the judges' data. Non-standard libraries may not be used in your solutions. You are also not allowed to include any files in your program. The following files will be included standard in all C programs:
   `stdio.h`, `math.h`, `ctype.h`, `strings.h`, and `string.h`.

3. Programming style is not considered in this contest. You are free to code in whatever style you prefer.

4. The time limit for all questions is the same and will be announced before the contest.

5. All questions regarding the contest material should be submitted to the judges through the `submit` command. You can ask the helpers in the room for any non-contest related matters or for getting your printer output. The helpers will not answer any questions regarding the contest material.

6. Judges' decisions are to be considered final. No cheating will be tolerated.

Good Luck!

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem A

Transferable Voting

In Fredonia, elections are conducted using a transferable vote system. When they enter the polling booth, Fredonians are presented with a list of candidates like

1. Tammy Fay Bakker
2. Jimmy Hoffa
3. Al Capone
4. Bonnie Parker
5. Elmer Fudd

A vote is cast by typing the numbers of the candidates in the preferred order. Thus a Fredonian who likes Jimmy Hoffa best, with Elmer Fudd second, but who does not care about the other three candidates, would type:

```
2 5
```

Each integer corresponds to the number preceding a valid candidate on the list. No number should appear more than once on a single ballot. Ballots which do not meet these rules are called *spoiled ballots*. Unspoiled ballots are called valid ballots. Your program should count these spoiled ballots. The outcome of the election should be the same as if the spoiled ballots had never been cast. When polling is complete, the outcome of the election is calculated as follows. First, all the first-preference votes (on valid ballots) are counted. After this count, if the number of votes for any candidate is more than half of the number of valid ballots, that candidate is declared the winner. Otherwise, the candidate(s) with the fewest number of votes is (are) eliminated. Ballot papers on which an eliminated candidate is mentioned are effectively modified by deleting that candidate, thereby "promoting" any lower-preference non-eliminated candidates. (A valid ballot from which all candidates have been eliminated remains a valid ballot.) If, after the elimination, there are any remaining candidates, the first-preference votes are counted again to determine a winner. If, after the elimination, no candidates are left, the election is declared indecisive. You are to write a program to calculate the outcome of Fredonian elections.

The input consists of a list of candidates and a list of ballots separated by a blank line (a newline by itself on an otherwise empty line). The list of candidates is an ordered list of up to 20 candidates, one per line. Each candidate's name can be up to 20 characters long. The first candidate's name will be preceded by the string "`1.  `", the second by "`2.  `", etc. The list of ballots consists of at most 1000 ballots. There are no blank ballots: each ballot has at least one integer. Each ballot consists of a sequence of integers, which are separated by single spaces, and is terminated by a newline. The list of ballots is terminated by an end-of-file.

Your program should produce the following output. As each candidate is eliminated, a message to that effect must be printed. If more than one candidate is eliminated in the same round (because they each had the minimum number of first-preference votes at that stage) then their eliminations should be output in the order in which the candidates originally were input. An elimination is recorded like this:

```
John Minor with 3 votes is eliminated.
```

Next your program should either declare the winner, like

```
The winner of the election is Clint Eastwood.
```

or declare that the election was indecisive by printing

```
The election was indecisive.
```

Finally, you should print a count of the valid ballots and the spoiled ballots. For example,

```
There were 457 valid ballots and 3 spoiled ballots.
```

After this last line there should be no blank line. The next two pages each contain a sample input and corresponding output.

Sample input 1:

```
1. Betty Boop
2. Elmer Fudd
3. Olive Oyl

1 2 3
1 2 3
1 2 3
2 1 3
2 1 3
2 1 3
3 1 2
3 1 2
3 1 2
```

Sample Output 1:

```
Betty Boop with 3 votes is eliminated.
Elmer Fudd with 3 votes is eliminated.
Olive Oyl with 3 votes is eliminated.
The election was indecisive.
There were 9 valid ballots and 0 spoiled ballots.
```

Sample input 2:

```
1. Bill Clinton
2. Barbara Bush
3. Margaret Thatcher
4. John Minor
5. Clint Eastwood
6. Jesse Ventura
7. Sonny Bono
8. Cher

2 4 6 8
1 2 3
5 7
8
6 6
5
5
5
5 7 2
8
3 2 5
```

Sample output 2:

```
John Minor with 0 votes is eliminated.
Jesse Ventura with 0 votes is eliminated.
Sonny Bono with 0 votes is eliminated.
Bill Clinton with 1 votes is eliminated.
Barbara Bush with 1 votes is eliminated.
Margaret Thatcher with 1 votes is eliminated.
The winner of the election is Clint Eastwood.
There were 10 valid ballots and 1 spoiled ballots.
```

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem B

Number Chains

Given a number, we can form a number chain by

1. arranging its digits in descending order
2. arranging its digits in ascending order
3. subtracting the number obtained in (2) from the number obtained (1) to form a new number
4. and repeat these steps unless the new number has already appeared in the chain

Note that 0 is a permitted digit. The number of distinct numbers in the chain is the *length* of the chain. You are to write a program that reads numbers and outputs the number chain and the length of that chain for each number read.

The input consists of a sequence of positive numbers, all less than 10^9, each on its own line, terminated by 0. The input file contains at most 10 numbers.

The output consists of the number chains generated by the input numbers, followed by their lengths exactly in the format indicated below. After each number chain and chain length, including the last one, there should be a blank line. No chain will contain more than 1000 distinct numbers.

Sample Input:

```
123456789
1234
444
0
```

Sample Output:

```
Original number was 123456789
987654321 - 123456789 = 864197532
987654321 - 123456789 = 864197532
Chain length 2

Original number was 1234
4321 - 1234 = 3087
8730 - 378 = 8352
8532 - 2358 = 6174
7641 - 1467 = 6174
Chain length 4

Original number was 444
444 - 444 = 0
0 - 0 = 0
Chain length 2
```

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem C

Count on Cantor

One of the famous proofs of modern mathematics is Georg Cantor's demonstration that the set of rational numbers is enumerable. The proof works by using an explicit enumeration of rational numbers as shown in the diagram below.

1/1	1/2	1/3	1/4	1/5	...
2/1	2/2	2/3	2/4		
3/1	3/2	3/3			
4/1	4/2				
5/1					

In the above diagram, the first term is 1/1, the second term is 1/2, the third term is 2/1, the fourth term is 3/1, the fifth term is 2/2, and so on.

You are to write a program that will read a list of numbers in the range from 1 to 10^7 and will print for each number the corresponding term in Cantor's enumeration as given below. No blank line should appear after the last number. The input list contains a single number per line and will be terminated by end-of-file. No more than 30 numbers will appear in the input file.

Sample input:

```
3
14
7
```

Sample output:

```
TERM 3 IS 2/1
TERM 14 IS 2/4
TERM 7 IS 1/4
```

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem D

Dining Diplomats

You have been hired to arrange the seating of diplomats at a dinner party. Your employer has invited 9 diplomats from several countries around the world. Each diplomat will speak from one to five languages. Diplomats not speaking a common language can not talk to each other. Furthermore, some of the countries represented will not have diplomatic relations with other nations represented, so these diplomats will not be speaking to each other either. Your task is to determine the seating arrangement at the dinner table for your employer and the 9 guests such that each person can speak to the person seated on each side.

The table to be used for dinner is round, and seats 10 people. Your employer will be seated at the first position at the table. The other positions will be numbered clockwise from 2 to 10. The person seated to the left of your employer is in seat number 2, the person seated to the right of your employer is in seat number 10.

Persons seated next to each other must speak a common language. A person may speak to the person on the left in one common language, and speak to the person on the right in another language.

The governments of the guests seated next to each other must have formally recognized each other. The government of your employer has diplomatic relations with the governments of all of his guests. The government of a guest may or may not have diplomatic relations with the governments of all the other guests.

Two diplomats from the same country will have diplomatic relations with the same list of countries. A country will always have diplomatic relations with itself. Diplomats from the same country may or may not speak any languages in common.

Input to the program will be a list of ten people, one per line, ended by end-of-file. The first line will represent your employer, the other lines will represent the guest diplomats. Each line consists of a sequence of words, where words are separated by a single space. Each word is a sequence of capitals. The first word consists of 3 characters and represents the code for the guest's country. The second word contains 1 to 5 characters and represents the languages that the guest speaks. Each language that the guest speaks is represented by a single letter. Finally, there is a list of up to 9 three-character words representing the codes of the countries of the other guests that this guest's government has diplomatic relations with.

Output of the program is a table of the seating arrangement. It contains one diplomat per line. Each line starts with the seat number followed by three words, all separated by single spaces. The first word is the language code for the person seated on the left. The second word is the country code for the diplomat. The third word is the language code for the person seated on the right. The seating arrangement should be printed in the order 1 to 10, and no blank line should appear after line 10. More than one solution may exist, but you need find only one. If no solution is found, output the sentence

```
NO SOLUTION EXISTS
```

Sample Input:

```
USA EF CHN GBR USR FRA FRG JPN ISR POR KOR
CHN CFE USA GBR FRA FRG
GBR ER USA CHN USR FRA FRG JPN ISR POR KOR
USR RF USA GBR FRA FRG
FRA F USA CHN GBR USR FRG JPN ISR POR
FRG ERG USA CHN GBR USR FRA JPN ISR POR
JPN JHG USA GBR FRA FRG JPN ISR POR KOR
ISR HER USA GBR FRA FRG JPN KOR
POR PGE USA GBR FRA FRG JPN
KOR KEC USA GBR USR JPN ISR
```

Sample Output

```
1 E USA E
2 E KOR E
3 E ISR H
4 H JPN G
5 G FRG G
6 G POR E
7 E GBR E
8 E USR F
9 F FRA F
10 F CHN E
```

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem E

Stamping Out Stamps

The mail clerk for Sirius Cybernetics just received a new postal scale which will display the cost (in cents) to send a given parcel. The postal scale is medium duty, and will only handle parcels requiring up to $29.99 in postage. She has requested you to write a program that will convert the postage into the number of stamps needed to cover the required postage such that the cost is minimized.

The mailroom stocks up to ten different types of stamps, and only ten stamps will fit on any given parcel.

Your program would read in a list of stamp values stocked by the mailroom (all separated by single spaces) followed by a sequence of amounts to convert (one amount per line). If it is impossible to reach the given amount exactly, you are to exceed the amount by as little as possible. If there are many sets of stamps that cover the amount for the same cost, you should pick the set with the least number of stamps. Output should consist of the given stamp values on a single line followed by a blank line, then, for each amount given, the amount that needs to be covered on a single line, followed by the stamps needed for the coverage on the next line, followed by a blank line. (So the last line of stamp values in the output is followed by one blank line.) As usual, all numbers are separated by single spaces. If no solution exists, print

```
NO SOLUTION EXISTS
```

The input file ends with an end-of-file. See next page for a sample input and output.

Sample Input

```
2 7 14 17 22 63 98
72
86
143
5
```

Sample Output

```
STAMP VALUES 2 7 14 17 22 63 98

AMOUNT 72
STAMPS USED 63 7 2

AMOUNT 86
STAMPS USED 63 14 7 2

AMOUNT 143
STAMPS USED 63 63 17

AMOUNT 5
STAMPS USED 2 2 2
```

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem F

Of(f) Course!

FAT CHANCE Airlines has had several of their aircraft stray off course within the last few months. After the last error nearly put an aircraft over the enemy's airspace, an investigation has shown that their navigation techniques do not take the wind speed and direction into account properly. In order to make life easier on their novice navigators, the airline has contracted with your team to produce a program that will provide the pilot with the proper heading for the aircraft, given a desired course and wind velocity.

For each flight segment, you will be given the desired course, the true air speed (the speed relative to the air) of the aircraft, the wind speed, and the direction the wind is blowing from. The pilot needs the heading to which she should steer the aircraft and the effective ground speed (the speed at which the aircraft is moving over the desired course relative to the ground). All speeds will be given in knots, and all directions will be given in degrees ($0^o \leq x < 360^o$, with 0^o = north, 90^o = east, 180^o = south, 270^o = west).

Input to your program will consist of a series of flight segments, one per line terminated by an end-of-file. Each line will consist of the wind speed, the wind direction, the desired course, and the true air speed, all given as free-format floating point numbers separated from each other by a single space.

Your program should produce for every line segment in the input file, including the last one, six lines followed by a blank line. Each of the six lines is labeled as in the example below. The first four lines reproduce the input numbers. The remaining two lines show the heading and effective ground speed calculated for each segment. Use a format analogous to the sample below. Each number must be accurate to 1/100th.

Sample Input

```
15.0 290.0 260.0 100.0
15.0 270.0 135.0 200.0
28.0 290.0 5.0 195.0
```

Sample Output

```
WIND SPEED 15.00
WIND DIRECTION 290.00
DESIRED COURSE 260.00
TRUE AIRSPEED 100.00
AIRCRAFT HEADING 264.30
GROUND SPEED 86.73

WIND SPEED 15.00
WIND DIRECTION 270.00
DESIRED COURSE 135.00
TRUE AIRSPEED 200.00
AIRCRAFT HEADING 138.04
GROUND SPEED 210.33

WIND SPEED 28.00
WIND DIRECTION 290.00
DESIRED COURSE 5.00
TRUE AIRSPEED 195.00
AIRCRAFT HEADING 357.03
GROUND SPEED 185.87
```

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem G

Double Trouble

The secret service of Hudonia has always had a strong appetite for strange numbers. This appetite seemed strongest when Hudonia was in trouble. The numbers they were looking for had the following property. Whenever you would right-rotate the number (that is, take away the last digit and put it in front of the number), you would end up with double the original number. Numbers possessing this property were called double-trouble numbers. For example, $X = 421052631578947368$ is a double-trouble number, since $2X = 842105263157894736$ which is a right rotation of X.

The number X is a double-trouble number in the number system with base 10. Any number system with base $p \geq 2$, however, has many such double-trouble numbers. In the binary number system (base $p = 2$), for example, we have the double-trouble numbers 01 and 0101. Notice that the leading zeros are necessary here in order to obtain the proper number after right rotation.

The secret service seemed to like the smallest double-trouble numbers in a number system most. For example, in the binary number system the smallest double-trouble number is 01. In the decimal ($p = 10$) number system, the smallest double-trouble number is 052631578947368421. Being a patriotic Hudonian, you are asked to write a program that computes for a given base p of a number system the smallest double-trouble number in that system.

The input file consist of a sequence of numbers, one per line, ended by end-of-file. Each number is less than 200. The output file consists of, for each number p in the input file, the smallest double-trouble number (including any necessary leading zeros) in the number system with base p. The double-trouble numbers are given in the format analogous to the example below and in the same order as in the input file. No blank line should appear at the end of the output. Digits in a base-p number system are given in decimal representation, and each digit (including the last one) is followed by a single space.

Sample input:

```
2
10
35
```

Sample output:

```
For base 2 the double-trouble number is
0 1
For base 10 the double-trouble number is
0 5 2 6 3 1 5 7 8 9 4 7 3 6 8 4 2 1
For base 35 the double-trouble number is
11 23
```

1993 East-Central Regionals
of the ACM International Collegiate Programming Contest

Problem H

Counting Patterns

Let n and k be numbers with $n > 0$ and $k \geq 0$. A *configuration* of the n-k-puzzle is an n-tuple with elements in the range $-k \ldots k$ such that their sum is zero. Configurations are considered equivalent when they can be obtained from each other by (a) cyclic permutation of the tuple over one or more positions, (b) reversal of the tuple, (c) sign reversal of all elements, or (d) combinations of (a), (b), and (c). Equivalence classes are called *patterns*.

For instance, $(0, 1, 1, -2)$ is a configuration of the 4-2-puzzle. Some equivalent configurations are: (a) $(1, -2, 0, 1)$, (b) $(-2, 1, 1, 0)$, (c) $(0, -1, -1, 2)$, and (d) $(-1, -1, 0, 2)$. Below is given a list of (the lexicographically largest) representatives of the 14 patterns of the 4-2-puzzle.

$(0,0,0,0)$	$(2,-2,2,-2)$	$(2,0,0,-2)$
$(1,-1,1,-1)$	$(2,-1,0,-1)$	$(2,1,-2,-1)$
$(1,0,-1,0)$	$(2,-1,1,-2)$	$(2,1,-1,-2)$
$(1,0,0,-1)$	$(2,0,-2,0)$	$(2,2,-2,-2)$
$(1,1,-1,-1)$	$(2,0,-1,-1)$	

Your program computes the number of patterns for a sequence of n-k-puzzles. The input consists of a sequence of pairs of integers n and k, which are separated by a single space. Each pair appears on a single line. The input is terminated by an end-of-file. The value for $n + k$ is at most 11. The output contains a sequence of integers, each on one line, representing the number of patterns for the corresponding n-k-puzzles in the input. No blank line should appear at the end of the output.

For example, the input

```
8 0
4 2
```

produces as output

```
1
14
```