ACM Hong Kong
Scholastic Programming Contest 1994

Total number of questions: 6

Total number of pages: 13

Contest time: 4 hours

1 Frequency Table of Bigram Analysis

Bigram analysis deals with statistical information about consecutive pairs of digits XY in decimal strings. We call X the *left* digit and Y the *right* digit. Given a decimal string, for each digit X that occurs in the string, bigram analysis finds the digit Y that follows X *most frequently* and computes Y's *percentage frequency*. The percentage frequency of Y with respect to X is calculated by dividing the number of occurrences of Y following X in the string by the number of times that X occurs as a left digit.

Consider the following string of 31 decimal digits:

5100999000999191919191919191919.

XY can either be 51, 10, 00, 09, 99, 90, 91, or 19. The frequency table for the example string generated by bigram analysis is shown in the following table.

X	F	Y	Percent
0	5	0	60%
1	10	9	90%
5	1	1	100%
9	14	1	64%

where

- X: The left digit of a pair.
- F: The number of times that X occurs as a left digit.
- Y: The digit that follows X most frequently. *In case of ties, the smallest digit is chosen.*
- Percent: The percentage frequency of the occurrences of Y, which is calculated by the occurrences of Y following X divided by F times 100. The percentage is *rounded* to the nearest integer.

Write a program to process input lines of decimal strings and generate frequency tables of the bigram analysis. Your program should *only* display the three digit pairs that give the three highest percentage frequencies of occurrence of the right digit(s). *In case of ties, the pair with the smallest left digit is chosen.* The output should show the three pairs in ascending order of the left digits. Moreover, statistical information should be accumulated so that cumulative statistical information can be used in the processing of the next input. The cumulative data should only be reinitialized when an input line consisting of the single character "`&`" is encountered. Note that if XY does not occur in the string, the percentage frequency of the occurrences of Y with respect to X is defined to be *zero*.

Input

When the program starts, it should prompt the user to enter the name of the input file with the display string "Enter input file name: ". Each line of the input file consists either a single "`&`" symbol or a string of decimal digits, the length of which is less than 81. You are **not** required to check for the validity of the input data.

Output

An input line of "&" serves only to reset the cumulative statistical information. It needs no computation and just displays the line "`Cumulative Statistical Information Reset.`" For each input line with a decimal digit string, your program should output four lines. The first line is simply an echo of the input string, followed by the "`:`" symbol. The next three lines contain the digit pairs XY that give the three highest percentage frequencies of occurrences of the right digit(s) respectively. Each output line is in the following form:

X `<SPACE>` Y `<TAB>` PF

where PF is the percentage frequencies of XY, `<SPACE>` is the space character, and `<TAB>` is the tab character.

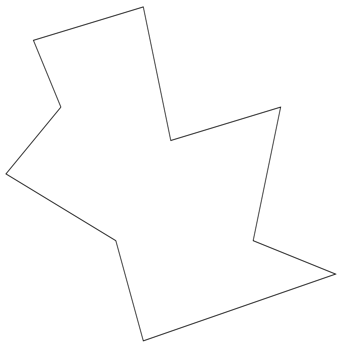
Examples

```
Input file:
  51009990009991919191919191919
  0909090909090909090
  &
  10

Output on screen:
  51009990009991919191919191919:
  1 9     90
  5 1     100
  9 1     64

  0909090909090909090:
  0 9     79
  1 9     90
  5 1     100

  Cumulative Statistical Information Reset

  10:
  0 0     0
  1 0     100
  2 0     0
```

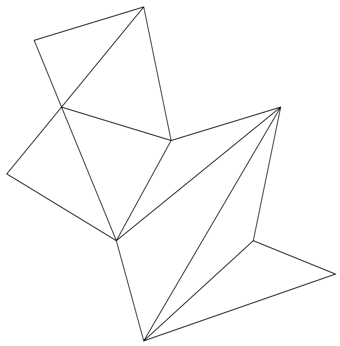
***** End of Question 1 *****

2 Triangulation

Triangulating a n-vertex polygon means adding diagonals between vertices in the interior of the polygon to subdivide the interior of the polygon into triangles. For example, the following figure shows a polygon and a triangulation of it. You are required to write a program to



A Polygon



A Triangulation of the Polygon

accept the specification of a polygon and then to compute and show a triangulation of it.

Input

When the program starts, it should prompt the user to enter the name of the input file with the display string “Enter input file name: ”. An input file specifies a polygon. The representation of a polygon is in the form of a sequence of coordinates of the vertices in clockwise order around the boundary. The sequence may begin at any vertex of the polygon. Each pair of coordinates in the sequence appears as two integers on a separate input line. The first integer is the x-coordinate and the second integer is the y-coordinate. The end of file character indicates the end of input data. The x-coordinate is in the range of 0 to 640, and the y-coordinate is in the range of 0 to 480.

Output

Your program should display on the screen in graphics the input polygon as well as the added diagonals to form a triangulation of the polygon. The shown triangulated polygon should clearly appear approximately at the centre of the monitor screen with its size large enough to observe all its edges and triangulated diagonals. The left lower corner of the screen is the origin (0,0). The x-axis is from left to right and the y-axis is from bottom to top.

Example

For example, the following input represents the polygon on the left of the figure above:

```
Input file:
 20 60
 40 80
 30 100
 70 110
 80 70
120 80
110 40
140 30
 70 10
 60 40
```

Your program should display the triangulated polygon on the screen similar to the one on the right of the figure above (page 3).

***** End of Question 2 *****

3 Resource Allocation

In a workshop, a team leader supervises a team of N workers to make some products. There are A available tools in the workshop, and a worker w_i $(1 \leq i \leq n)$ can request up to M_i tools in maximum to make his product. After he has made his product, he releases all tools he holds back to the team leader. One major task of the team leader is to make appropriate allocation of the tools to the workers so that all workers will accomplish their jobs without a deadlock. More precisely, the team leader must ensure that the current allocation state is in a *safe state* in which there exists a *safe allocation sequence* to the tools.

A sequence of resource allocations to workers, $< W_1, W_2, ..., W_n >$ (*i.e.*, a permutation of w_i), is a safe allocation sequence for the current allocation state if, for each worker W_i, the tools that W_i can still request (*i.e.*, current needs C_i), can be satisfied by the currently available tools plus the resources held by all the worker(s) W_j, with $j < i$. If the tools that worker W_i needs are not immediately available, then W_i can wait until all W_j have finished. When they have finished, W_i can obtain all of his needed tools, complete his job, return his allocated tools, and take a rest. When W_i finishes, worker W_{i+1} can obtain his needed tools, and so on. If no such a sequence exists, then the current allocation state is said to be *unsafe*.

Given the current allocation state of the workshop, you are required to write a program to help the team leader to determine if the current allocation state is safe or unsafe. If it is safe, the program also prints out a safe allocation sequence.

Input

When the program starts, it should ask the user to enter the filename of an input file. Values of N and A, and the current allocation state for the N workers are specified in the input file, the format of which is as follows:

1. The first line contains an integer which is the number of workers N in the workshop $(1 \leq N \leq 16)$.
2. The second line contains an integer which is the maximum number of available tools A in the workshop $(1 \leq A \leq 100)$. Note that A includes those tools which have allocated to the workers.
3. Each of the lines from Line 3 to Line $N + 2$ of the file contains two integer numbers which correspond to the maximum needs M_i and current needs C_i of the worker w_i $(1 \leq i \leq N)$.

Output

If the state is a safe allocation state, the program should output

```
The given allocation state is safe.
A safe allocation sequence is .........
```

and if the state is an unsafe allocation state, it should output

```
The given allocation state is unsafe.
```

Example

Input file:

```
3
12
10  5
4   2
9   7
```

Output on screen:

```
The given allocation sate is safe.
A safe allocation sequence is w2, w1, w3.
```

***** End of Question 3 *****

4 Cyclic Redundancy Code

Cyclic redundancy code (CRC) is a well-known coding technique for error detection in network communications. It is based on treating bit strings as representations of polynomials with coefficients of 0 and 1 only. A k-bit frame is regarded as the coefficient list for a polynomial with k terms, ranging from x^{k-1} to x^0. For example, bit string 110001 has 6 bits and thus represents a six-term polynomial $x^5 + x^4 + x^0$.

When the CRC technique is employed, a sender and its corresponding receiver must agree upon a **generator polynomial**, $G(x)$, in advance. Both the high-order and low-order bits of the generator must be 1. To compute the **checksum** for some frame with m bits, corresponding to the polynomial $M(x)$, the frame must be longer than $G(x)$. The basic idea is to append a checksum to the end of the frame $M(x)$ so that the transmitted frame is divisible by $G(x)$. When the receiver gets the checksummed frame, it tries to divide it by $G(x)$. If there is a remainder, there has been a transmission error.

The arithmetic used in CRC calculation is in modulo 2. There are no carries for addition or borrows for substraction. Both addition and subtraction are identical to the EXCLUSIVE-OR operation. Division is preformed in regular steps. For examples,

- $10011011 + 11001010 = 01010001$
- $11110000 - 10100110 = 01010110$
- $11010110 \div 10011 = 1100$ (remainder 00010)

To compute a transmitted frame with a proper CRC checksum, the following steps can be followed:

1. Let r be the degree of $G(x)$. Append r zero bits to the low-order end of $M(x)$ so that the resulting polynomial becomes $x^r M(x)$.
2. Divide the bit string of $G(x)$ into the bit string of $x^r M(x)$ using modulo 2 division.
3. Subtract the remainder (which is always r or fewer bits) from the bit string of $x^r M(x)$ using modulo 2 substraction. The result is the checksummed frame to be transmitted. We call this polynomial $T(x)$.

For example, given $G(x) = 10011$ and $M(x) = 1101011011$, the transmitted frame $T(x) = 11010110111110$.

In this question, you are required to write a program to generate the transmitted frame with CRC checksum and to check if a received frame has a transmission error.

Input

When the program starts, it should ask the user to enter the filename of an input file. Each line of the input file states a request for CRC computation.

If the first character of a line is '`S`', it is a checksum computing request, and the next two fields, each of which is separated by a blank character, are the bit strings of $M(x)$ and $G(x)$ respectively.

If the first character of a line is 'R', it is a checksum checking request, and the next two fields, each of which is separated by a blank character, are the bit strings of a received frame $T'(x)$ and $G(x)$ respectively.

You program should handle all requests until end-of-file is encountered. The maximum length of bit string $T(x)$ is 128. The maximum degree of $G(x)$ is 16.

Output

For a checksum computing request, it should compute the transmitted frame $T(x)$ from the given $M(x)$ and $G(x)$. All $M(x)$, $G(x)$ and $T(x)$ bit strings should appear in the output.

For a checksum checking request, it should check if the received frame $T'(x)$ has a transmission error or not. All $T'(x)$, $G(x)$, and the checking result should appear in the output. In case there is an error, the remainder should be printed out.

All requests must be numbered in the output display. Moreover, your program should check illegal inputs, such as invalid request character, and the bit string of G(x) being longer than that of M(x) etc.

Example

Input file:

```
S 11100110 11001
R 111001100110 11001
R 111001101111 11001
```

Output on screen:

```
(1) M(x) = 11100110, G(x) = 11001
    T(x) = 111001100110

(2) T'(x) = 111001100110, G(x) = 11001
    correct

(3) T'(x) = 111001101111, G(x) = 11001
    error (remainder 1001)
```

***** End of Question 4 *****

5 Transportation of Chemicals

A daily routine of the ABC chemical company headquarter is to ship inventory supplements to its branches. Extreme care must be taken in the transportation arrangement since some chemicals are *incompatible* with one another in the sense that placing them in the same compartment can cause explosion. Therefore, normal transportation operation involves grouping compatible chemicals and placing them in the same container truck. Unfortunately, all except one truck of the company breaks down today due to a typhoon. With only one truck in service, the company would like to ship as many different kinds of chemicals to its branches as possible to maximize cost effectiveness. In other words, the task at hand is to find a maximal (i.e. the largest possible in size) subset of n chemicals that are compatible. We call the subset a *maximal compatible set*. Note that there can be more than one maximal compatible subset.

As the senior analyst programmer of the ABC company, you are to implement a program to solve the maximal compatible subset problem as specified. Suppose that the chemicals are numbered 0 to $n-1$. The input to your program is an $n \times n$ matrix M, where M is defined in such a way that $M(i,j) = 1$ if chemical i is incompatible with chemical j and $M(i,j) = 0$ otherwise, where i and j are in $\{0, \ldots, n-1\}$. Note that M is symmetric. Your program should output the size of the maximal compatible sets and give the elements of one maximal compatible set.

Input

When the program starts, it should prompt the user to enter the name of the input file with the display string "Enter input file name: ". The first line of the input file contains a positive integer n that specifies the number of chemicals. The rest of the input file contains n lines of n single digits of 0's and 1's separated by space. The jth digit on row i represents the entry $M(i,j)$ of matrix M. You may assume that the test data will not contain matrices larger than 21×21.

Output

Your output should consist of two lines. In the first line, your program should print "`Size of maximal compatible set:`" followed by the size of the maximal compatible sets. In the second line, you program should print "`A maximal compatible set:`" followed by the elements of a maximal compatible set. The elements should be in ascending order and separated by space. **Note that your program should be able to find and print a solution within 2 minutes.**

Example

```
Input file:
  10
  0 1 0 0 1 0 0 0 0 1
  1 0 0 1 0 0 1 0 0 0
  0 0 0 0 0 0 0 1 0 1
  0 1 0 0 0 1 0 0 0 0
  1 0 0 0 0 0 0 0 1 0
  0 0 0 1 0 0 0 0 0 1
  0 1 0 0 0 0 0 0 0 0
  0 0 1 0 0 0 0 0 1 0
  0 0 0 0 1 0 0 1 0 1
  1 0 1 0 0 1 0 0 1 0

Output on screen:
  Size of maximal compatible sets: 5
  A maximal compatible set: 0 2 3 6 8
```

***** End of Question 5 *****

6 Stern-Brocot Number System

The Stern-Brocot (SB) tree was discovered independently by Moriz Stern, a German mathematician, and Achille Brocot, a French clockmaker. The construction of the SB tree starts with the two fractions $0/1$ and $1/0$, and *repeatedly* inserts $(m+m')/(n+n')$ between adjacent fractions m/n and m'/n'. It is easy to verify that the following inequalities hold:

$$\frac{m}{n} < \frac{m+m'}{n+n'} < \frac{m'}{n'}.$$

For example, the first step gives us one new entry between $0/1$ and $1/0$,

$$0/1, 1/1, 1/0;$$

and the next gives two more:

$$0/1, 1/2, 1/1, 2/1, 1/0.$$

The next gives four more,

$$0/1, 1/3, 1/2, 2/3, 1/1, 3/2, 2/1, 3/1, 1/0;$$

and then we shall get 8 more, 16 more, and so on. The resulting array can be arranged as an infinite binary tree, the top levels of which are shown in Figure 1.

The construction of the SB tree guarantees that all positive reduced fractions are generated and they occur exactly once in the tree. This is the basis of a new number system for rational numbers. We consider traversing the tree from the root $(1/1)$ to reach a reduced fraction a/b. If we use the letter `L` to stand for walking down the left branch and the letter `R` for walking down the right branch, then a string of `L`'s and `R`'s, which we call an *SB string*, uniquely identifies a node on the tree. The fraction $1/1$ corresponds to the empty string, ϵ.

To implement the Stern-Brocot number system on electronic computers with only finite amount of resources, we need to impose an upper bound n on the number of levels of the SB tree. If a fraction appears in the top n levels of the SB tree, then the fraction can be represented by an SB string of length less than n. In case a fraction does not appear in the top n levels of the SB tree, the tree allows us to identify the *smallest* interval containing the fraction, the bounds of which are from the top n levels of the tree. For example, if we restrict the SB tree to 4 levels, the number $7/10$ is best approximated by the interval $(2/3, 3/4)$. We use the SB string `LRRL*` to denote the interval, where `*` signifies that the string represents an approximation. The symbol `*` must occur only as the last character of an SB string of length n.

Write a program to read in a series of records. The first record is a positive integer n less than 23, which specifies the number of levels of the Stern-Brocot tree used. Each of the subsequent records contains either an SB string or a fraction a/b expressed in the form `f(a,b)`. If the input is a legal SB string, print out its value in reduced fractional form `f(a,b)` or the smallest interval of fractional bounds in the form `i(f(a,b),f(a',b'))` that contains the value, where appropriate. Otherwise print the string "`OUT OF RANGE`" instead. If the input is a fraction, print the SB string that represents or approximates the fraction, and in addition the smallest interval that encloses the fraction if the fraction cannot be represented exactly. We use `e` to denote the empty SB string.

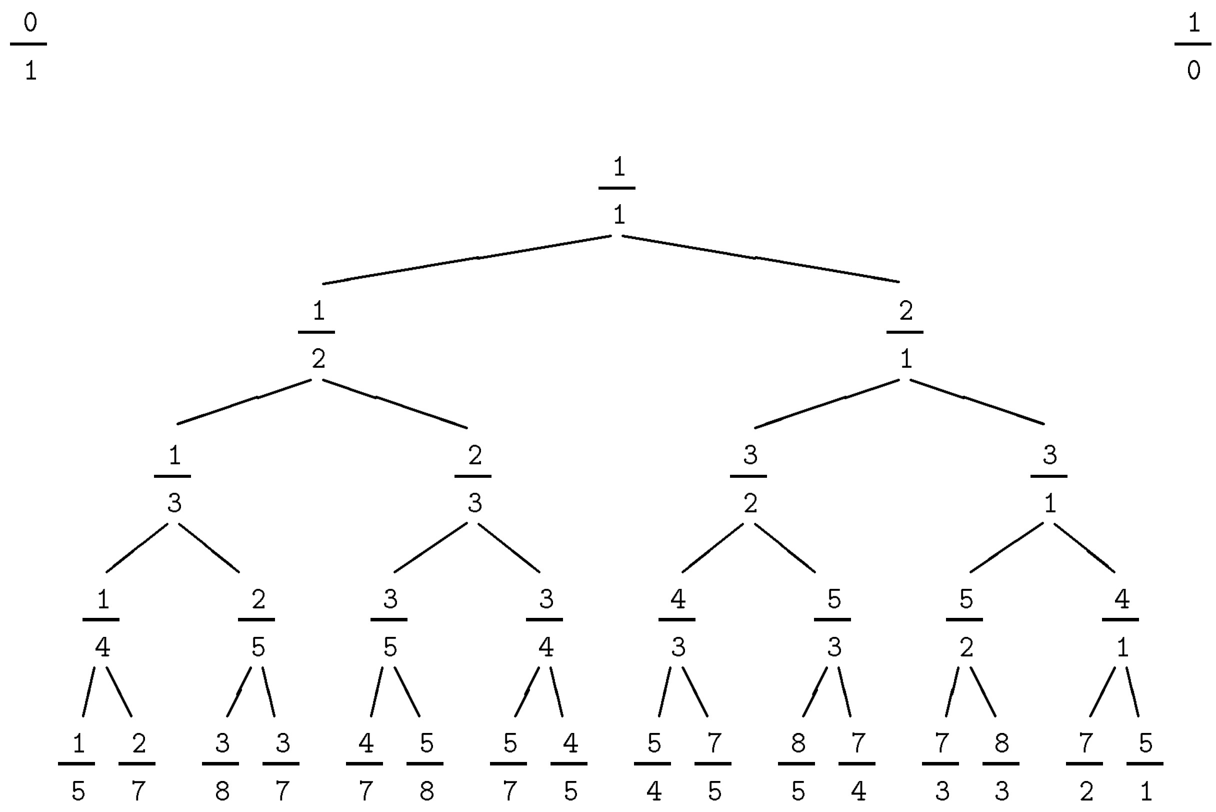


Figure 1: Top levels of the Stern-Brocot tree.

Input

When the program starts, it should prompt the user to enter the name of the input file with the display string "Enter input file name: ". The first line of the input file contains a positive integer less than 23. Subsequent lines contains either (1) a fraction a/b of the form `f(a,b)`, where `a` and `b` are positive integers, or (2) a string made up of letters `L`, `R`, and `*`. You are **not** required to check for the syntax of the input records.

Output

For each input line, your program should echo the input, followed by an "`=`" symbol and the number in its alternative representation. If the input is a fraction that cannot be represented exactly using an SB string, then the SB string approximation should be followed by "`in`" and the smallest interval that enclose the fraction. An interval `[a/b,a'/b']` is printed as `i(f(a,b),f(a',b'))`. *Note that all output fractions must be in their reduced form.* Some input data may not make sense even if it is syntactically correct. In that case, print "OUT OF RANGE" in place of the alternative representation.

Examples

Input file:

```
5
f(7,10)
LLLL
LRLRL*
f(5,3)
L*RL
```

Output on screen:

```
f(7,10) = LRRLL* in i(f(2,3),f(5,7))
LLLL = f(1,5)
LRLRL* = i(f(3,5),f(5,8))
f(5,3) = RLR
L*RL = OUT OF RANGE
```

***** End of Question 6 *****