FIU-UCF Programming Contest

February, 16 1991

Problem Name	Filename
Key to Success	KEY
Wetlands of Florida	WETLAND
Nasty Virus	VIRUS
Magic Numbers	NUMBERS
Simultaneous Equations	EQUATIONS
Raucous Rockers	ROCKERS

Call your program file:	*Filename*.PAS
	or *Filename*.C
Call your input file:	*Filename*.IN
	or as discussed in the problem text.

All file names are assumed to bein lowercase

Key to Success

Any one-to-one mapping, f, of any alphabet to itself can be used to encode text by replacing each occurrence of any letter, c, with f(c). One such mapping could be the mapping of a letter to three positions beyond the letter in the alphabet. That is, a→d, b→e, c→f, d→g and so on. With this mapping, "The car is blue." will be encoded as "Wkh fdu lv eoxh." Write a program that decodes the contents of its input file according to the following guidelines:

1. Only letters are encoded. Letters are mapped to letters. Uppercase letters are different from their lowercase counter parts.

2. The mapping that defines the encoding is one-to-one. That is, two different letters never map to the same letter of the alphabet (a→x and t→x is impossible).

3. There are two input files – the input file, KNOWN.IN, contains a text (not encoded) and the input file, ENCODED.IN, contains an encoded text. This text is to be decoded by your program.

4. Both KNOWN.IN and ENCODED.IN are written by the same person.

5. It is to be assumed that any person uses letters of the alphabet with the same RELATIVE FREQUENCY from document to document and no two letters are used with the same frequency. That is, the most frequently used letter in KNOWN.IN maps to the most frequently used letter in ENCODED.IN; the second most frequently used letter in KNOWN.IN maps to the second most frequently used letter in KNOWN.TXT and so on.

Sample Input:

```
KNOWN.IN:
     abacxbacac
ENCODED.IN
     qqqqqrrrrssstt
```

Sample Output:

```
     aaaaaccccbbbxx
```

Wetlands of Florida

A construction company owns a large piece of real estate within the state of Florida. Recently, the company decided to develop this property. Upon inspection of the property, however, it was revealed that the land, at various locations, contained bodies of water. This came as a shock to the owners of the company, for they were from out of state and not familiar with wetlands of Florida. The situation was very grave and the owners not knowing that such bodies of water can be converted to beautiful lakes that will increase the value of the land around them, were about to abandon the construction project. Fortunately, this fact was brought to the owners' attention by a smart FIU graduate who worked for the company and consequently the construction project started.

The engineers divided the construction site by a grid into uniform square cells such that each square cell entirely contained either water or land. (How they did it, of course, is anybody's guess.) Now, the question that the engineers are to answer is the following: "Given the row and column number of a grid cell that contains water, what is the area of the lake containing that cell." (an area is measured by number of grid cells it contains.)

You are to write a program to answer this question!

The input consists of $0 < n \leq 99$ lines each containing $0 < m \leq 99$ character long sequence of "L"s and "W"s followed by $k > 0$ lines each containing a pair of integers i and j. The first n lines will represent the n by m grid covering the land where a "W"/"L" at the cth character of the rth line indicates water/land within the cell at row r and column c of the grid. The pairs of integers on the last k lines, each represent the row and column numbers of some grid cell that contains water.

The output for each pair of integers, i and j, on the last k lines of input, consists of an integer,on a separate line, indicating the area of the lake containing the grid cell, at row i and column j of the grid.

Sample input:

```
LLLLLLLLL
LLWWLLWLL
LWWLLLLLL
LWWWLWWLL
LLLWWWLLL
LLLLLLLLL
LLLWWLLWL
LLLLWLLLL
LLLLLLLLL
3 2
7 5
```

Sample output:

```
12
3
```

Nasty Virus

Our programmers have been recently experiencing a strange phenomenon. Some old FOR-TRAN programs that have been working for years suddenly stopped working!! A close inspection of the situation revealed the existence of a nasty virus in our computer. The virus wakes up every night, and adds random lines, some of them valid FORTRAN statements, to all FORTRAN programs. (The existing lines are not changed.) Furthermore, it was discovered that:

1. the virus may insert lines **ONLY BEFORE** the last line and after the first line;

2. the virus may insert an assignment to a variable "v" in one of two ways:

   The original program does not contain any assignment to "v" – in this case the new assignment to "v" can be inserted anywhere in the original program. (before the last line)

   The original program does contain one or more assignments to "v" – in this case the new assignment to "v" can be inserted only after the last assignment to "v" in the original program. (before the last line)

You are to write a program that will read one corrupted FORTRAN program and restores the program to its original state.

Fortunately, the original FORTRAN programs compute values of certain functions by using ONLY assignment statements where the right-hand side expression of each assignment statement is an expression over the operators: +, *, /, - and operands {a, b, ... y, z, 0, 1, ... 9}. Furthermore, our programs contained **NO USELESS** assignment statements (an assignment to a variable, v, is useless if the value of v is not consequently used in the program) and the last assignment statement of the program computed the value of the desired function. Also, our smart programmers always start their program by a line of comment, "Cn", where where n is the number of lines (not counting the comment line itself) contained in the program.

The input consists of one possible corrupted FORTRAN program containing only assignment statement as described above.

The output, consists of the corrupted FORTRAN program in its original state.

Sample input:

```
C5
                x=5+6

                a=x+7

                LLWWLLWLL

                d=x+7

                if (x-a)10,3,2

                c=x+b

                e=2*x

                a=x*7

                kljhkjh kjhkjhk kjh

                y=x*a+c+d
```

Sample output:

```
C5
                x=5+6

                a=x+7

                d=x+7

                c=x+b

                y=x*a+c+d
```

Magic Numbers

Write a program that finds and displays all pairs of integers s_1 and s_2 such that:

1. neither s_1 nor s_2 have any digits repeated; and

2. $s_1/s_2 = N$, where N is a given integer;

The input consists of one line of input containing N.

The output consists of a sequence of zero or more lines each containing $s_1/s_2 = N$, where s_1, s_2 and N are the integers described above.

Sample input:

```
1234567890
```

Sample output:

```
1234567890 / 1 =   1234567890
2469135780 / 2 =   1234567890
4938271560 / 4 =   1234567890
6172839450 / 5 =   1234567890
8641975230 / 7 =   1234567890
9876543120 / 8 = 1234567890
```

Simultaneous Equations

Write a program that will solve a n by n system of simultaneous equations where the coefficients of the equations are complex numbers. (Recall that a complex number is an imaginary number of the form $a + b * \sqrt{-1}$, where a and b are real numbers.)

The input consists of $0 < n \leq 99$ lines each containing $n + 1$ pairs of complex numbers in the form (a, b). The $j^{th}, 1 \leq j \leq n$, complex number at line i is the coefficient of the jth unknown in the ith equation and the last complex number at line i represents the right-hand side of the ith equation.

The output consists of n lines containing pairs of the form (a, b). The pair on line i of output represents the ith root of the input system of equations. In case the input system of equations can not be uniquely solved, your program should produce no output.

Sample input:

```
(1,0) (2,0) (3,0) (4,0)
(2,0) (3,0) (4,0) (20,0)
(3,0) (4,0) (5,0) (26,0)
```

Sample output:

```
(1,0)
(2,0)
(3,0)
```

Sample input:

```
(1,0) (2,0) (3,0) (4,0)
(2,0) (4,0) (6,0) (8,0)
(3,0) (4,0) (5,0) (26,0)
```

Sample output:

Raucous Rockers

You just inherited the rights to n previously unreleased songs recorded by the popular group Raucous Rockers. You plan to release a set of m compact disks with a selection of these songs. Each disk can hold a maximum of t minutes of music, and a song can not overlap from one disk to another. Since you are a classical music fan and have no way to judge the artistic merits of these songs, you decide on the following criteria for making the selection:

1. The songs will be recorded on the set of disks in the order of the dates they were written.

2. The total number of songs included will be maximized.

The input consists of one line containing the values of n, t and m (integer numbers) followed by a line containing a list of the length of n songs, $t_1, t_2, ... t_n$ ordered by the date they were written (Each t_i is less than t minutes long and $sum_{i=1}^{n} t_i > m \times t$.)

The output, consists of one integer indicating the number of songs that, following the above selection criteria will fit on m disks.

Sample input:

```
10 5 3
3, 5, 1, 2, 3, 5, 4, 1, 1, 5
```

Sample output:

```
6
```