1993 YSU-ACM High School Programming Contest

Problem #1

Pascal Source File: P1.PAS **BASIC Source File: B1.BAS** **Data file: D1.DAT**

Description: A children's puzzle that was popular 30 years ago consisted of a 5 by 5 frame which contains 24 small squares of equal size. A unique letter of the alphabet was printed on each small square. Since there were only 24 squares within the frame, the frame also contained an empty position which was the same size as a small square. A square could be moved into that empty position if it were immediately to the right, to the left, above or below the empty position. The object of the puzzle was to slide squares into the empty position so that the frame displayed the letters in alphabetical order. The illustration below represents a puzzle in its original configuration and in its configuration after the following sequence of 6 moves:

1. The square above the empty position is moved.
2. The square to the right of the empty position is moved.
3. The square to the right of the empty position is moved.
4. The square below the empty position is moved.
5. The square below the empty position is moved.
6. The square to the left of the empty position is moved.

T	R	G	S	J
X	D	O	K	I
M		V	L	N
W	P	A	B	E
U	Q	H	C	F

T	R	G	S	J
X	O	K	L	I
M	D	V	B	N
W	P		A	E
U	Q	H	C	F

Input: Input for your program consists of several puzzles. Each is described by its initial configuration and a sequence of moves on the puzzle. The first 5 lines of each puzzle description are the starting configuration. The next line gives the sequence of moves.

The first line of the input corresponds to the top line of squares in the puzzle. The other lines follow in order. The empty position in a frame is indicated by a blank. Each input line contains exactly 5 characters, beginning with the character on the leftmost square (or a blank if the leftmost square is actually the empty frame position). The input will correspond to a legitimate puzzle.

The sequence of moves is represented by a one line sequence of As, Bs, Rs, and Ls to denote which square moves into the empty position. A denotes that the square above the empty position moves; B denotes that the square below the empty position moves; L denotes that the square to the left of the empty position moves; R denotes that the square to the right of the empty position moves. All input sequences will be valid and will not move outside of the bounds of the puzzle. No line will be longer than 80 characters.

Input will be terminated with 5 consecutive lines of **00000** in the puzzle description.

Output: Output for each puzzle begins with an appropriately labeled number (Puzzle #1, Puzzle #2, etc.) The final configuration of the puzzle should then be printed. Format each line for a final configuration so that there is a blank character between two adjacent letter. Treat the empty square the same as a letter. Separate output from different puzzle records by at least one blank line.

Example:

```
Input:
  TRGSJ
  XDOKI
  M VLN
  WPABE
  UQHCF
  ARRBBL
  ABCDE
  FGHIJ
  KLMNO
  PQR S
  TUVWX
  RAAALLLL
  00000
  00000
  00000
  00000
  00000
```

```
Output:
  Puzzle #1
    T R G S J
    X O K L I
    M D V B N
    W P   A E
    U Q H C F

  Puzzle #2
      A B C D
    F G H I E
    K L M N J
    P Q R S O
    T U V W X
```

1993 YSU-ACM High School Programming Contest

Problem #2

Pascal Source File: P2.PAS **BASIC Source File: B2.BAS** **Data file: D2.DAT**

Description: Display the perimeter of an n by n square.

Input: A sequence of nonnegative integers n, one per line, terminated by 0.

Output: The perimeter of a square with sides n. An * should be used to denote the side of a square. Format each line so that **there is a blank character between adjacent *s**. Separate output from different squares by at least one blank line.

Example:

```
Input:          Output:
  3               * * *
  5               *   *
  0               * * *

                  * * * * *
                  *       *
                  *       *
                  *       *
                  * * * * *
```

1993 YSU-ACM High School Programming Contest

Problem #3

Pascal Source File: P3.PAS **BASIC Source File: B3.BAS** **Data file: D3.DAT**

Description: Consider the string "AAAABCCCCCDDDD" consisting of alphabetic characters only. This string is of length 14. Since the string consists of alphabetic characters only, duplicate characters can be removed and replaced with a duplication factor n. With this technique the string can be compress and represented by "4AB5C4D". The compressed string is of length 7. This technique is considered *simple data compression*. Write a program which takes a series of strings in compressed form and recreates the original uncompressed string.

Input: A series of strings, one per line. Strings will be of the format "nA..." where n, the duplication factor, is an integer between 2 and 99 and A is an uppercase alphabetic character. A string may contain single characters not prefixed with a duplication factor. If this were not the case, for instance, the string "AABCDE" would be compressed to "2A1B1C1D1E". To avoid this, single characters will not be prefixed with a duplication factor. The string "AABCDE" would be compressed to "2ABCDE". Input will be terminated with **00000** in the first 5 positions of a new line. The maximum length of an input string is 80 characters.

Output: The uncompressed strings, 40 characters per line (it may be necessary to break an uncompressed string over multiple lines).

Example:

```
Input:              Output:
 3A4B7D              AAABBBBDDDDDDD
 4AB5CD              AAAABCCCCCD
 22D7AC18FGD         DDDDDDDDDDDDDDDDDDDDDDAAAAAAACFFFFFFFFFF
 00000               FFFFFFFFGD
```

1993 YSU-ACM High School Programming Contest

Problem #4

Pascal Source File: P4.PAS **BASIC Source File: B4.BAS** **Data file: D4.DAT**

Description: A group of n survivors and a monkey are stranded on a deserted island in the Caribbean. The survivors gather all of the coconuts on the island, m, together into one pile. That night while they are sleeping the first survivor wakes up and decides to take his portion of the coconuts. He divides the coconuts into n equal piles and hides one of them. He puts the other $n-1$ piles back into one large pile, takes one of those coconuts, and gives it to the monkey. The first survivor then goes back to sleep. Each survivor then wakes up and follows the same routine. Determine how many coconuts remain after all of the survivors are done.

Input: Sequences of nonnegative integers n and m one per line terminated with $n = 0$ and $m = 0$.

Output: The number of coconuts remaining.

Notes: Any fractional share should be rounded down. For instance, if there are 14 coconuts and 5 survivors the first survivor's share will be 2.

Example:

```
Input:                Output:
  3                     30
  106
  0
  0
```

The first survivor takes his share of the coconuts, 35, and gives one to the monkey, leaving 70. The second survivor takes his share of the coconuts, 23, and gives one to the monkey, leaving 46. The third survivor takes his share of the coconuts, 15, and gives one to the monkey, leaving 30.

1993 YSU-ACM High School Programming Contest

Problem #5

Pascal Source File: P5.PAS **BASIC Source File: B5.BAS** **Data file: D5.DAT**

Description: The ABC Company wants to determine the comparative efficiency of operations in several plants. To do so, they need to know the cost of labor for a one week period. The plants where the study is being held have four types of employees. Type 1 employees receive time and a half for hours over 35 per week. Type 2 employees receive time and a half for hours over 40 per week. Type 3 employees receive double time for hours over 30 per week. Type 4 employees receive straight time for all hours (the same pay no matter how many hours are worked.) You are to determine the total cost of labor and total hours for each plant.

Input: The input records will be of the following form:

Position	Contents
1-6	Employee Identification Number
7	Plant Code (0-9)
8	Type Code (1-4)
9-11	Hours worked this week to 1 decimal place
12-16	Hourly wage to 2 decimal places

Input is terminated by **000000** in the employee number.

Output: Output will consist of lines for each plant. Each line will contain plant code, total cost of labor and total hours for the plant. You may round your output to the nearest cent if you want. Separate output from different plants by at least one blank line.

Notes: Hours worked and hourly wage will not contain a decimal point; for instance, 12.3 hours worked will be represented as 123 in the data file. Similarly, an hourly wage of $4.50 would be 00450. Your output should be in dollars and hours and should contain a decimal point.

Example:

```
Input                                Output
    0001951242501250                    1    936.88    68.5
    0002221426001500
    0053962137500850                    2    459.88    66.5
    0062712329000450
    0000000000000000
```

1993 YSU-ACM High School Programming Contest

Problem #6

Pascal Source File: P6.PAS **BASIC Source File: B6.BAS** **Data file: D6.DAT**

Description: The factorial of an integer n, written $n!$, is the product of all the integers from 1 to n, inclusive. The factorial quickly becomes very large; 13! is too large to store as an integer on most computers, and 35! is too large for a floating-point variable. Your task is to find the rightmost non-zero digit of $n!$. For example, $5! = 1 \times 2 \times 3 \times 4 \times 5 = 1\underline{2}0$, so the rightmost non-zero digit of 5! is 2. Also, $7! = 1 \times 2 \times 3 \times 4 \times 5 \times 6 \times 7 = 50\underline{4}0$, so the rightmost non-zero digit of 7! is 4.

Input: Sequence of integers n, one per line, $0 \leq n \leq 100$. Input is terminated by $n = 0$.

Output: For each n print n and the rightmost non-zero digit of $n!$.

Example:

```
Input:       Output:
  3            The rightmost non-zero digit in   3! is 6
  5            The rightmost non-zero digit in   5! is 2
 10            The rightmost non-zero digit in  10! is 8
  1            The rightmost non-zero digit in   1! is 1
100            The rightmost non-zero digit in 100! is 4
  0
```