1982 NATIONAL SCHOLASTIC PROGRAMMING CONTEST

CONTEST PROBLEMS

February 10, 1982

(Revised)

GENERAL INSTRUCTIONS: In writing programs for the contest problems which follow, all input is to be read from unit 5 and all output is to be written to unit 6. Program output must begin at the top of a page. Unless otherwise stated, you may assume all input data are valid and that the exact formatting of output is at your discretion.

1982 NATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 1: Banking

This program is to print a detailed log of checkbook transactions. Input to the program is a sequence of data items, each containing an amount, a transaction code, and a date. The items are in chronological order by date. Input records after the first represent checks written and deposits made. Each check is represented by an input record having the form

xxxxx.xxDBmmddyy (in columns 1 - 16)

where the x's are digits of the amount, DB is the transaction code for a check, and mmddyy is month, day, and year of the date the check cleared. Each deposit is represented by an input record having the form

xxxxx.xxCRmmddyy

with the obvious meaning.

Your program is to print a report showing all transactions and service charges (explained below). For each entry, print one line in the following form:

PREVIOUS BALANCE xxxxxx.xx$\begin{matrix}\text{CR}\\\text{DB}\end{matrix}$ xxxxx.xx$\begin{matrix}\text{CR}\\\text{DB}\\\text{SC}\end{matrix}$ NEW BALANCE xxxxxx.xx$\begin{matrix}\text{CR}\\\text{DB}\end{matrix}$

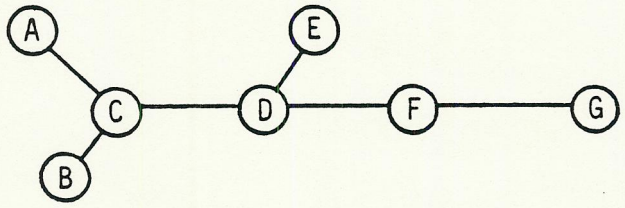
No leading + or - signs are used in these lines. Balances less than zero are printed followed by DB and balances greater than or equal to zero are printed followed by CR. All values are to be printed with at least one digit to the left of the decimal point; thus, 3¢ should appear as 0.03DB. If the balance of the account falls below $500.00 any time during the month, a service charge of $0.12 per check cleared anytime that month is assessed against the account. The service charge (SC) is to be printed and charged to the account before listing any transactions for the next month. No SC line is to be printed if no checks are cleared or if the balance remains high. At the end of the data, you are to print any service charge for the final month. Note that being below minimum balance at the end of a month implies being below minimum balance at the beginning of the next month.

The first input record contains the initial balance (sign indicated by DB or CR) and the date on which accounting is to begin. This record should not be treated as a transaction.

1982 NATIONAL SCHOLASTIC PROGRAMMING CONTEST

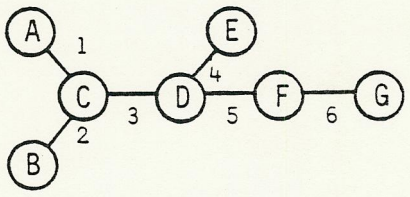
Problem 2: Network Messages

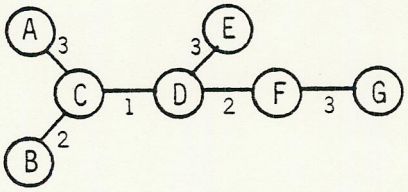
A computer network has been constructed linking large computers in different cities. Not all computers are directly connected, but each can send a message to any other by passing it through zero or more intermediate computers. The network is a tree, meaning that there is exactly one path from any computer (perhaps through some others) to any other computer. An example of such a network is shown below:



It is desired to determine how to broadcast a message from one particular computer to all others in the network in minimum time. It takes one unit of time for a message to be sent from one computer to another, and a computer may send a message to only one other at any given time. During the first unit of time, the originator of the message can send it to one neighboring computer. During the second unit of time, both the originator and that neighbor can send the message to respective neighbors. At this point, up to four computers have seen the message. During the third unit of time, each of these can send the message to another neighbor, etc.

Since each computer may have several neighbors, the choice of neighbor to which to send the message next can greatly affect the total time to broadcast the message. For example, in the diagrams below, the numbers on the edges indicate the unit of time in which that edge carries the message. The node labeled C is the originator of the message in each case, but in the diagram on the left, six units of time are required, while in the diagram on the right only three units of time are required.





Your program must do the following:

1. Read in the description of a network as described below.
2. Read in the originating node of the message.
3. Determine an optimal order of sending the message. Print that order, showing the sender and receiver of each transmission for each unit of time.

Problem 2

As an example, the network shown above should generate output equivalent to the following:

```
TIME 1
  C             TO D


TIME 2
  C             TO B
  D             TO F


TIME 3
  C             TO A
  D             TO E
  F             TO G
```

A network will be represented by input data structured as follows:

line 1:	column 1:	n = number of nodes, $2 \leq n \leq 9$
lines 2 ... n+1:	column 1:	node number (single digit)
	columns 3..15:	name of city
	column 17:	number of first neighbor
	column 19:	number of second neighbor
	⋮	
	column xx:	number of last neighbor
	column xx+2:	0 (marker of end of neighbor list)
line n+2:	column 1:	number of originator node

For example, the network above is described by:

```
7
1 A             3 0
2 B             3 0
3 C             1 2 4 0
4 D             3 5 6 0
5 E             4 0
6 F             4 7 0
7 G             6 0
3
```

There will be exactly four such sets of data to be processed. Note that answers may be non-unique.

1982 NATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 3: Breaking a Cipher

A message has been enciphered using a simple substitution cipher. This means that a permutation of the alphabet has been chosen and each letter of the original message has been replaced by the corresponding letter of the permuted alphabet. For example, using the permutation:

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

the message THE CAT IS OUT OF THE BAG becomes the message SAD RJS QF WVS WZ SAD XJT.

The frequencies of occurrence of letters in English text is not uniform; the letter E occurs most often, followed by T, etc. Given a sufficiently large enciphered message, the frequencies of occurrence of the letters should indicate which letters are represented. For example, if the letter D occurred most often in the enciphered message, we would expect it to represent the letter E, as it does in the above example.

Your program must decipher an enciphered message using this idea. The data consist of two messages. The first message will be of normal English text (unenciphered). Your program should use this text to determine the relative frequencies of the letters in normal English. The second message will be an enciphered message. Your program should determine the relative frequencies of occurrence of the letters in this message, then decipher the message by substituting for each occurrence of the k-th most frequent letter, the k-th most frequent letter of plain English text. Print out the deciphered message.

Input consists of up to 20 lines of plain text, then a line with an asterisk in column 1, then up to 20 lines of enciphered text, then another asterisk in column 1. Spaces and punctuation will not be enciphered. All text will be uppercase only.

Input lines contain 80 characters, some of which may be blanks.

1982 NATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 4: Budget Roofers

Budget Roofers, Inc. sells its services through door-to-door salesmen who can measure the roof accurately enough but have been a disaster at estimating material needs. The company needs a program which will estimate the minimum number of bundles of shingles which are needed for each job. Unfortunately, the data are recorded in a strange manner which management is unwilling to change because it took so long to get the salesmen to do it in any predictable fashion.

For each roof, there is one data card in the following layout:

Columns	Format	Content
1-5	I5	Job Number (not zero)
7	I1	Roof Type
9-13	F5.1	Roof slope, in Percent
15-20	F6.3	Dimension A
22-27	F6.3	Dimension B
29-34	F6.3	Dimension C (type 3 only)
36-41	F6.3	Dimension D (type 3 only)

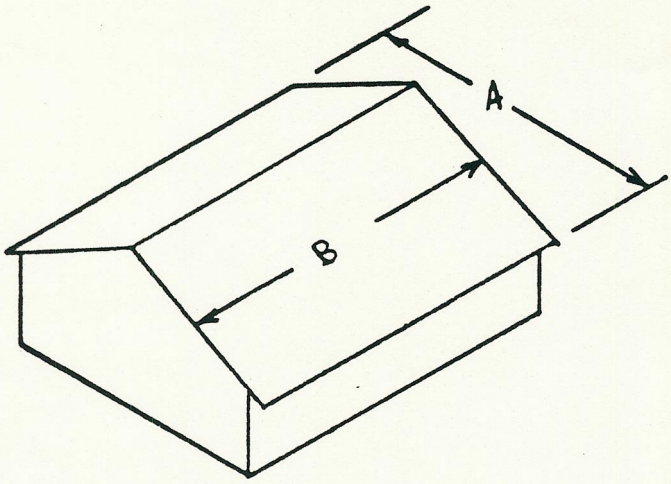
The deck is unordered except that the cards for a multi-roof job will bear the same job number and be contiguous in the deck. A blank card terminates the deck. Roof types are illustrated in the figures below. All dimensions are measured horizontally and are recorded using the convention that in the representation

ff.iie

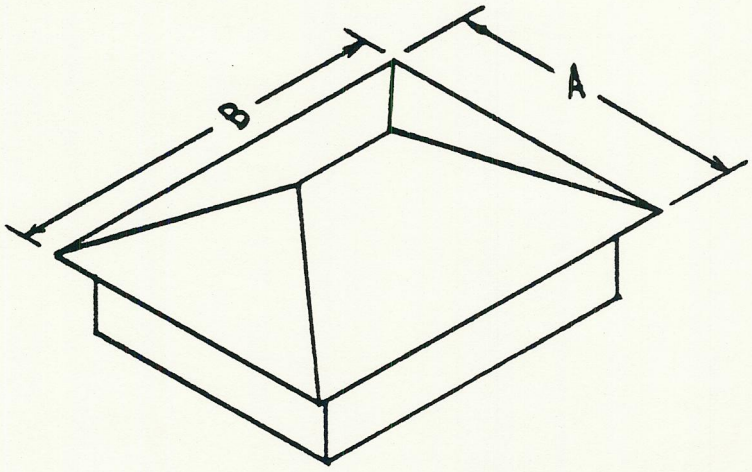
ff is the feet portion of the measurement, ii is the inches portion (and is less than 12) and e (in the range 0-7) denotes eights of an inch.

The slope of a roof is the same on all portions of the roof. It is dimensionless and is 100 times the ratio of rise to horizontal run.

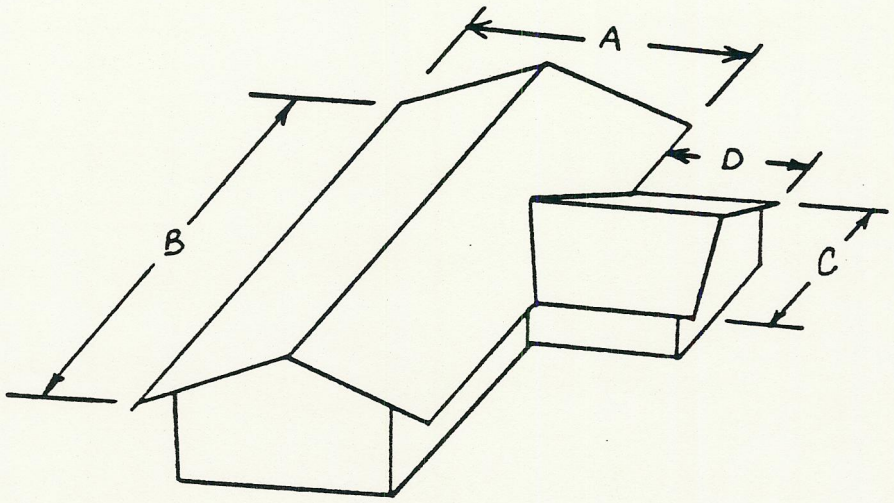
It takes 3 bundles of shingles to cover 100 ft^2 of roof surface. The needed program is to produce a two-column report showing each job number and the integral number of bundles of shingles needed for that job. When there are multiple roofs on a single job, all shingles will be the same weight and color for that job, so a fraction of a bundle left over on one roof on that job can be used on another roof of the same job.



Type 1



Type 2



Type 3

Constraint for all roofs: $B \geq A \geq C$

1982 NATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 5: Intersections

You are to write a program to determine all points with integer X- and Y-coordinates within the intersection of areas enclosed by the boundaries of two-dimensional geometric figures. For example, in figure 1, the points

(0,1), (0,2), and (-1,-1)

lie strictly within the intersection of the interiors of the circle, triangle, and rectangle. (Points on figure boundaries are to be excluded.) Your program processing this figure should print out the list of points given above in some reasonable format. Order of points is unimportant. Note that the list may be empty.

The geometric figures which may occur are rectangles, triangles, and circles, none of which may be degenerate and all of which will be correctly specified. The interiors of as many as three shapes may be intersected. Your program should process any number of collections of figures. All figures lie within the square bounded by the points (-10,10), (10,10), (10,-10), (-10,-10). Before listing the intersection points for each group of figures, your program should echo the input in order to specify the figures involved.

The arrangement of input data is given below:

	Card #	Columns	Format	Value
repeated any number of times	1	1	I1	N(number of shapes, 1≤N≤3)
	2...N+1	1	A1	S(shape code--see below)
		2-17	4(2I2)	(X,Y)coordinates (see below)

The final input card is blank and terminates the program. The shape code is defined as follows:

1) S='R' represents a rectangle. There will be four (X,Y) pairs representing the corners, given in clockwise or counterclockwise order.

2) S='T' represents a triangle. There will be three (X,Y) pairs.

3) S='C' represents a circle. The first (X,Y) pair represents the center and the next number (in cols. 6-7) represents the circle radius.

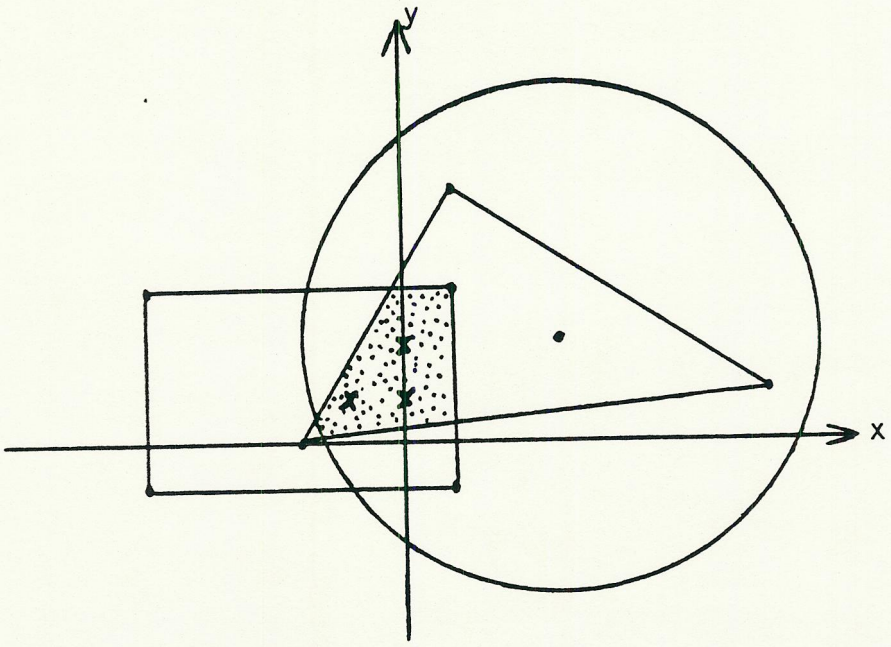


Figure 1.

1982 NATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 6: Ole the Cobbler

Ole, the cranky cobbler, works according to the following schedule. Whenever a customer comes into his shop, he decides how long the job will take. He always estimates perfectly. If the job will take over five minutes, he puts the job at the end of his job queue. If it will take five minutes or less, he puts it at the front of his job queue (even if there is another short job already at the front of his queue). Of course, he always finishes the job he is working on before he starts a new one.

The flow of jobs into Ole's shop over a week's time is described on a deck of 80-character records. The data were gathered during a typical 5-day week, Monday through Friday. The records, in format (I3,6X,I2) are in the order in which the corresponding jobs arrived at the shop. For each job record, the first value is time-of-day measured in minutes from the time the shop opened each day and the second is the number of minutes of work the job requires, always positive. (Since Ole never opens before 8:00 a.m., the three digits for minutes will suffice.) Each day's jobs are terminated by a blank card in the deck. Ole never gets more than 50 jobs per day and always gets at least one. He stays at the shop until each day's jobs are finished before going home to supper. Any jobs left on his doorstep are encoded as arriving at minute zero, his opening time, and more than one job can arrive in the same minute.

Ole is considered cranky because he is so particular about the way he handles these things. His shop has a large clock with a sweep second hand which he uses to severely dicipline his procedures. If a job arrives while Ole is busy, he enqueues it and continues. If a job arrives when he is idle, he waits until the beginning of the next minute to start on it, so all jobs arriving in that minute are enqueued before one is selected to work on. (Jobs are not waiting until the end of their arrival minute.) When a job is being finished, Ole works on it until the end of its final minute, enqueueing jobs arriving in that minute. In other words, his inflexible order of business within each minute is:

1) If idle, start the next job on the queue.
2) Enqueue all arriving jobs.
3) Finish job in process if due this minute.

You are to write a program which will calculate the average delay time for jobs brought into the shop each day. The average delay time in minutes is the average time that a job is in the shop before Ole starts to work on it. Print a table in the following format:

DAY	AVERAGE DELAY
MONDAY	XXX.X
TUESDAY	XXX.X
WEDNESDAY	XXX.X
THURSDAY	XXX.X
FRIDAY	XXX.X

For example, if a 4-minute job and a 6-minute job arrive at 8:05 and a 5-minute job arrives at 8:12, the schedule of events is

8:06 Start 4-minute job (waiting time 0 for job)
8:09 Finish 4-minute job
8:10 Start 6-minute job (waiting time 4 for job)
8:12 Enqueue 5-minute job
8:15 Finish 6-minute job
8:16 Start 5-minute job (waiting time 3 for job)

not used

1982 NATIONAL SCHOLASTIC PROGRAMMING CONTEST

Problem 8: Microcomputer Switch Controller

You are to write a program that could hypothetically run on a microcomputer that could control 16 switches around a home. Your program can exercise the following functions relative to the switches: 1) Turn any or all on, 2) Turn any or all off, 3) Sense the state of a switch. The home owner is to have the following capabilities relative to the program: 1) Command any or all switches to be turned on, 2) Command any or all switches to be turned off, 3) Query the state of any or all switches, 4) Specify a time of day for command (1) or command (2) to take place each day.

Your program does not have available a time-of-day clock so time will be simulated through input. Times are in 24-hour form, from 0000 to 2300. If you read a "TIME" card that says it's 0400 (4:00 a.m.), then it's 4:00 a.m. until you encounter a new "TIME" card. Time can't go backwards so if 0400 is followed by 0100, that implies that 21 hours have elapsed. Times are 4 digits, with minutes (the last 2 digits) always zero.

The input comes from data cards. The input is keyword-oriented in free format. Keywords are TIME, ON, OFF, ?, ALL, AT, STOP. Keywords and operands may appear anywhere on the input card, separated by at least one blank. Only one command appears on a card. Command forms are (where # is a decimal number between 1 and 16, representing a switch number, and τ is a 4 digit number representing a time):

# ON	switch # on immediately
# OFF	switch # off immediately
# ON AT τ	switch # on at time τ every day
# OFF AT τ	switch # off at time τ every day
? #	print current setting of switch #
? ALL	print current setting of all switches
TIME τ	advance simulated time to τ
STOP	halt simulation; print message

Your program should echo all input, as well as printing messages from ? and STOP commands.

Example

Input	Meaning	Action to be Taken
1 ON	Turn on Switch #1	None
TIME 0700	It is now 7:00 a.m.	None
?1	What is the state of Switch #1?	Print SWITCH 1 ON
2 ON AT 1300	Turn on Switch 2 at 1:00 p.m.	None
TIME 1300	It's now 1:00 p.m.	None
?ALL	What is the state of all 16 switches?	Print 1 2 3 4 16 ON ON OFF OFF ... OFF
STOP	simulation halts	Print SIMULATION END

The command # ON AT τ is to take effect as time advances to τ.

If a switch is to be automatically turned on or off at a particular time, e.g., 7 ON AT 1000, then its state is set to ON every time 10:00 a.m. rolls around unless it is explicitly cancelled by a 7 OFF AT 1000. When an ON cancels an OFF (or an OFF cancels an ON) in this way, nothing occurs at 1000--the switch is not changed in any way. To change the system from automatic ON at 1000 to automatic OFF at 1000, two 7 OFF AT 1000 commands are needed.

You may assume all commands are properly entered. There will not be more than 100 automatic ON and OFF commands in effect at any time. All switches are initially off and time is 0000.