1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

CONTEST PROBLEMS

February 16, 1983

GENERAL INSTRUCTIONS: This booklet contains six problems. For any problem, you may write a FORTRAN or Pascal program. You cannot receive credit for the same problem more than once, however. FORTRAN programs should read from unit 15 and write to unit 16. Pascal programs should read from file INPUT and write to file OUTPUT. Unless otherwise stated, programs may assume that input data are correct, that values in numeric fields are right-justified, and that input values representing zero contain explicit digits. Good luck!

1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

PROBLEM 1: MOUSETRAP

Mousetrap is a card game which is played as follows. A set of cards marked with the numbers 1, 2, 3, ..., n is dealt in any order, face upwards, in the form of a circle. The player begins at the first card and counts around the circle in the direction the cards were dealt. If the k-th card has the number k on it--which event is called a hit--the card is removed from the circle and the player begins counting again from 1.

For example, if a pack of only 4 cards is used, and these cards come in the order 3214, then the player would obtain the second card 2 as a hit, next he would obtain 1 as a hit, but if he went on forever, he would not obtain another hit. On the other hand, if the cards in the pack were initially in the order 1423, the player would obtain successively all four cards in the order 1, 2, 3, 4.

Assume that you are playing with a deck of nine cards. You are to write a program which will determine the hits that would be obtained with a particular ordering of the cards. The input to the program will be an unspecified number of records, each of which specifies an ordering of the cards. Each record will have the first card's number in column 1, the second card's number in column 3, etc., through column 17. For each record, you should print the input, followed by a list of all the hits that would be obtained. The hits may be indicated in any convenient format.

SAMPLE INPUT:

```
1 3 5 7 9 2 4 6 8
9 7 3 1 2 4 5 6 8
```

RESULTING OUTPUT:

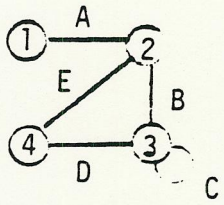
```
INPUT  1 3 5 7 9 2 4 6 8

     HIT 1
     HIT 8

INPUT  9 7 3 1 2 4 5 6 8

     HIT 3
     HIT 1
     HIT 7
```

1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

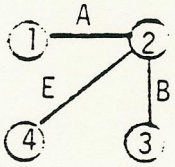
PROBLEM 2: SPANNING TREES

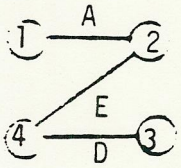
A graph G is a set of nodes N, together with a set of edges E. An edge is a connection between two (not necessarily distinct) nodes. A connected graph is one in which any node is reachable from any other by traversing a sequence of edges. A spanning tree T for a connected graph G is a graph whose set of nodes is N (the same as for G)and whose set of edges, E', is a subset (not necessarily proper) of E. Furthermore, T must be a connected graph with no circuits, that is, there is one and only one path from any node to any other node. E' is thus a minimal subset of E such that T is connected. A spanning tree is not necessarily unique.

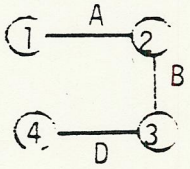
As an example, the graph shown below contains four nodes (1-4) and five edges (A-E):



For this graph, there are three possible spanning trees:







You are to write a program to generate spanning trees from graphs. The graphs may contain up to 15 nodes. Your program should process any number of input graphs, printing the edges of one spanning tree for each graph.

Input to your program comes in sets. The first record in a set contains a value for n, the number of nodes in graph G, right justified in columns 1-2. This record is followed by n records, each representing one node and its connections. The first of these records represents node number 1. The record representing a node contains n values specifying the edges (connections) between this and other nodes. Each of these values appears right-justified in a 2-column field--a 0 indicates no connection and a 1 means a connection exists. For the record representing node k, columns 1-2 contain a 0 or a 1 indicating whether there is an edge from node k to node 1; columns 3-4 indicate whether there is a connection from node k to node 2; columns (2j-1)-(2j) indicate whether there is a connection from node k to node j. There is no separator between successive graph descriptions.

For each input set representing a graph, you are to print, one pair per line, a list of pairs of nodes identifying the edges of a spanning tree for the graph. Each edge should be represented exactly once in this list.

SAMPLE INPUT (for graph shown above):

```
4
0 1 0 0
1 0 1 1
0 1 1 1
0 1 1 0
```

SAMPLE OUTPUT:

```
THE CONNECTIONS ARE

1 - 2
2 - 3
2 - 4
```

1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

PROBLEM 3: TABLE MANIPULATION

This problem is an exercise in organizing a one-dimensional list of numbers into a 2-dimensional table with a variable number of columns. The numbers must be organized into a table which reads by columns, left to right. The lengths of the columns should be balanced so that all rows except the last are filled. The empty spaces in the last row should be at the right end of the row.

EXAMPLE:

LIST: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17

COLUMNS: 5

TABLE:

1	5	9	12	15
2	6	10	13	16
3	7	11	14	17
4	8			

Write a program to read a collection of records (< 10,000) each with a positive integer in columns 1-5. A zero value terminates the list. A second set of records should be read next, with various positive integer values for C, the number of columns, in columns 1-5. After each value of C is read, the sum of the elements in each row of the table should be computed and printed. Sums will be less than 9 digits. The 2-dimensional table itself should not be printed.

Using the sample data shown above for various values of C, output similar to the following should be produced:

```
C    ROW SUMS

5    42 47 52 12
************************************
1    153
************************************
2    11 13 15 17 19 21 23 25 9
************************************
```

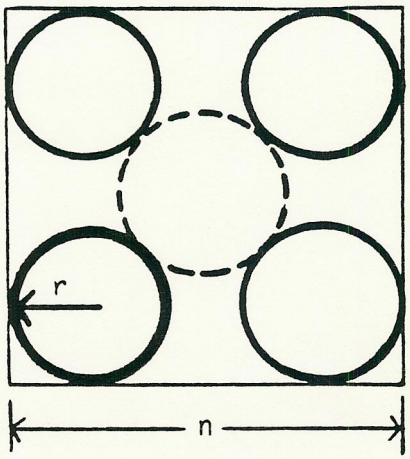
Either vertical or horizontal printing of row sums is acceptable, but a line of asterisks should separate output generated for different values of C. A zero value for C should cause the program to terminate.

1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

PROBLEM 4: INSCRIBED FIGURES

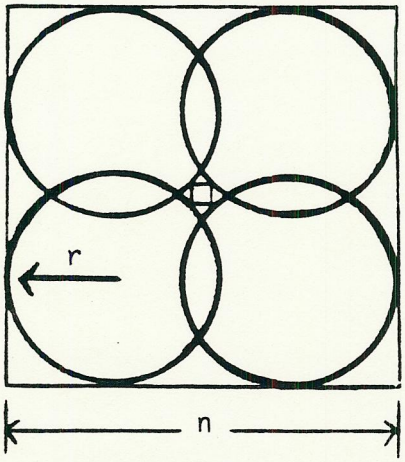
Consider the following configurations of geometric shapes:

(a)



shape = CIR

(b)



shape = SQR

Write a program that will process records containing values having the following meanings:

n	-	the length of a side of a square
r	-	the radius of 4 circles, each located in a corner of the square and tangent to two sides (see above figures)
shape	-	CIR for circle, SQR for square

Based on the value of shape, your program should calculate the area of the largest inscribed square (having sides parallel to those of the grid) or circle that lies completely within the large square and intersects the four circles in at most four points.

The input data format is given below (an arbitrary number of input records is to be processed):

Columns	Value
1 - 3	n (right-justified integer, $0 \leq n \leq 100$)
4	blank
5 - 10	r (real, positive, right-justified number of the form xxx.xx)
11	blank
12 - 14	shape (CIR or SQR)

For each input record, print n, r, and shape, appropriately labeled. Also print the area of the inscribed figure. This value must be correct ± 0.1.

1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

PROBLEM 5: DATA DICTIONARY

One important component of many information systems today is a data dictionary, which defines the characteristics of data elements in data records. This problem requires that a simple data dictionary (a list of all data elements and their forms) be used to validate the contents of input transactions and to generate a data exception report.

The input file contains two sets of data--a set containing dictionary definitions and a set containing input transactions. The first set contains no more than 15 records and is delimited with a record containing 00 in columns 1-2. These records are in ascending order by starting position (see below). Each record in this set is 80 characters long, describes one data element, and is formatted as follows:

Columns	Values
1-2	Starting position p of field in transaction record ($0 < p \leq 80$), integer
4-5	Field length f ($1 \leq f \leq 80$), integer
7	Field type: = 1 numeric (contains digits, signs, periods, or blanks) = 2 alphanumeric (contains any characters) = 3 date in the form MMDDYY (This field must contain exactly six digits, but may include embedded blanks anywhere. Also, $1 \leq MM \leq 12$ and $1 \leq DD \leq 31$.)
9	Presence: = 1 required field (must not be blank) = 2 optional field
11-30	Data element name (used only for printing)
31-80	Unused

You may assume that all dictionary data are correct and consistent. Your program is not responsible for checking characters in fields not specified by the data dictionary.

The second set of data contains transactions to be validated with the dictionary rules provided in the first data set. The format of these records is dependent on the definitions in the dictionary. However, the record length is known to be 80 characters. End-of-file will signal the end of the second set of data.

A report of invalid data records should be generated in the following format:

ERRONEOUS TRANSACTIONS

```
XXX ..................................XXX
            message 1
            message 2
               .
               .
               .

XXX ..................................XXX
            message 1
            message 2
            message 3
               .
               .
               .
```

where XXX...XXX represents an offending transaction and message i is composed of the data element name from the dictionary followed by brief English text describing the nature of the error. Representative error messages are:

```
SSNO          IS MISSING
SALARY         IS NOT NUMERIC
BIRTHDAY        HAS MONTH > 12
NETPAY    CONTAINS AN EMBEDDED BLANK
```

Note: For numeric fields, numbers must be correctly formed, but they need not be right-justified. For dates, assume all months contain 31 days.

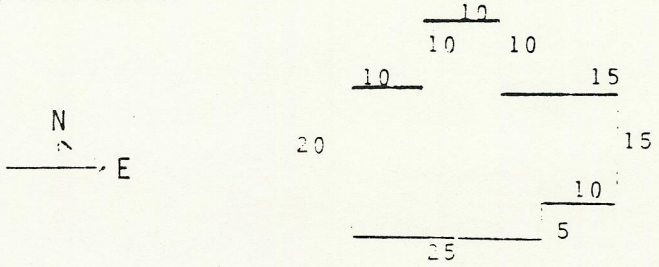
1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

PROBLEM 6: LOG CABINS

HYPOXIA Properties, Ltd. is in the business of selling log cabin homes. Their charges are based on the square footage of the particular model being ordered and whether HYPOXIA provides the labor or not. Although their log home construction and log cabin kits are very good, HYPOXIA has some difficulty in calculating areas because floor plans come in many shapes and sizes.

John Hardwood, the company's founder, would like to have a computer program to assist in this calculation. The company estimator would like to be able to enter data describing the cabin's dimensions, log diameters and lengths, number of stories desired, etc., and get back the square footage of the cabin (rounded to the nearest foot) and the cost of the log cabin (in dollars).

Floor plans are encoded by giving, starting at an arbitrary corner, a list of successive (direction, distance) pairs which define the log cabin perimeter, where directions are always given as N, S, E, or W, and dimensions are always integral feet. For example, a cabin with the following dimensions



might be coded as (N, 20), (E, 10), (N, 10), (E, 10), (S, 10), (E, 15), (S, 15), (W, 10), (S, 5), (W, 25).

Log cabin prices are calculated according to the following rules:

(1) The "base charge" for a log cabin kit (using 6" diameter logs) is $20.00 per square foot. If HYPOXIA provides both the kit and the labor, a base charge of $35.00 per square foot is charged (once again for 6" diameter logs).

(2) 8" diameter logs are available for an additional 10% of the base charge described above.

(3) A second story (loft floor) is available for an additional 15% of the base charge calculated in (1) or (2) above.

(4) The standard base rate is figured using an average log length of 8 feet. The customer may specify log lengths of 10 feet or greater for an additional 5%.

Program input can describe an arbitrary number of log cabins. Each log cabin description requires two input records as follows:

	Columns	Field
1st record	1	default log length (T for 8' average, F for 10')
	2	log diameter (6" (6) or 8" (8))
	3	blank
	4	number of floors (1 or 2)
	5	blank
	6	HYPOXIA to provide labor (T for yes, F for customer will build)
2nd record	1-2	number of corners in cabin (not greater than 18)
	3	blank
	4	direction (N, S, E, or W)
	5-6	distance
	7	blank
	8	direction
	9-10	distance
	11	blank
	:	:

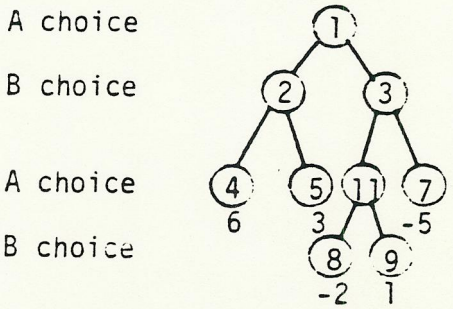
For each two records read, the program should print the area of the described floor plan and the calculated cost (accurate to the nearest dollar).

— Not used

1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

PROBLEM 8: BINARY GAME TREES

A binary game tree is a binary tree in which each node has either 0 or 2 descendants. Each leaf of the tree has a data value associated with it. The two descendants from any internal node represent the outcomes of making and of not making a particular move, and the value at a leaf of the tree is a measure of the helpfulness of the moves along the path from the root to that leaf in winning. The tree represents moves for two players; the root and all nodes an even distance from the root are moves for player A to select, while all nodes an odd distance from the root are moves for player B. An example is shown below.



Player A seeks to maximize the value of moves selected, while player B seeks to minimize the benefit to A by selecting the lower-valued path. At any node representing a choice for A, A will select the higher value; at any node representing a choice for B, B will select the lower value. The value of an internal subtree is thus either the maximum or the minimum value of its subtrees, depending on whether the original subtree is for player A or B, respectively.

Input to your program will be the representations of several binary game trees. You are to determine the value (for A) of each tree. Your output is to be the value of the tree; you do not need to print the entire tree. A tree may have at most fifteen nodes. Data for this problem are arranged in sets. Each set contains one to fifteen records; the first record describes the root (node number 1), and each subsequent record describes the next higher-numbered node. Each record contains three fields containing right-justified values:

Columns	Field
1 - 3	Node number of left offspring ($1 \leq n \leq 20$)
4 - 6	Node number of right offspring ($1 \leq n \leq 20$) The above fields contain zeroes if this node is leaf.
7 - 10	Value of the leaf ($-99 \leq v \leq 99$) or zero if this node is not a leaf.

Each set is terminated by a record having -1 in the left offspring field and zeroes in the other fields.

SAMPLE INPUT (for the tree shown above):

```
 2  3   0
 4  5   0
11  7   0
 0  0   6
 0  0   3      (Note that node number 6 is not used.)
 0  0   0
 0  0  -5
 0  0  -2
 0  0   1
 0  0   0      (Node number 10 is not used.)
 8  9   0
-1  0   0
```

The program is to continue processing trees until reaching end-of-file.

Not need

1983 ACM NATIONAL SCHOLASTIC PROGRAMMING CONTEST

PROBLEM 7: POLYNOMIAL MULTIPLICATION

A polynomial $P(x) = a_0x^0 + a_1x^1 + a_2x^2 + \ldots + a_nx^n$ can be represented by a collection of ordered pairs where the first element of the pair is the coefficient and the second element is the exponent of x. For example, the polynomial $P(x) = 7 + 3x^2 + 4x^9 + 17x^{21}$ might be represented by (7,0), (3,2), (4,9), (17,21).

You are to write a program which will read an unspecified number of pairs of records, each of which contains a representation of a polynomial. The program should then print the ordered pairs corresponding to the product of the two polynomials. The exponents of the pairs should be in strictly increasing order. (Input polynomials are similarly represented.) Print out the description of one polynomial for each pair of input records.

Assume all coefficients are integers. Each input polynomial will contain nine or fewer terms. The input coefficients and exponents are all non-negative and less than 100. The number of terms in the polynomial will be right-justified in columns 1-2. The coefficient of the first term will be right-justified in columns 3-6, the exponent of the first term right-justified in columns 7-10, the coefficient of the second term in columns 11-14, etc.

SAMPLE INPUT:

```
 3   3   0  16   3  23   5
 2   2   2  10   5
```

RESULTING OUTPUT:

COEFF	EXP
6	2
62	5
46	7
160	8
230	10

representing $6x^2 + 62x^5 + 46x^7 + 160x^8 + 230x^{10}$