2003 APICS Programming Contest

The 2003 Atlantic Canada programming Contest was held at University Prince Edward Island in Charlottetown PEI Friday Oct 17 2003. There were 19 teams of three students from 10 universities in the four Atlantic provinces.

Each team of 3 students was given 6 problems to try to solve in 5 hours using one computer following the rules for the ACM programming contests.

Each problem has sets of associated data and matching output files. The judges had additional files.

a. Spreadsheet - A Sedgwick
b. Getting in Line
c. Cheap Fares - A Sedgwick
d. Magic Decoder Ring - Martin van Bommel
e. Urban Elevations
f. Breaking Down Is Hard To Do - Todd Wareham

The participating teams came from the following universities:

```
Acadia University, Wolfville N.S.
Dalhousie University, Halifax N.S.
St Marys University, Halifax  N.S.
St Francis Xavier University, Antigonish N.S.

Mount Allison University, Sackville N.B.
Universite de Moncton, Moncton N.B.
University of New Brunswick, Fredericton Campus.
University of New Brunswick, St John Campus.

University of Prince Edward Island, Charlottetown.

Memorial University of Newfoundland, St. John's.
```

The top 4 teams from this competition advance to the next level of competition in Rochester N.Y.

Spreadsheet Calculator

Consider the following 3 lines of sample input representing a spreadsheet:

```
10      0       -10     =A0+B0+C0
11      12      13      =A1-B1-C1+114
=A0+A1  =B1+B0  =C0+C1  =A2+B2+C2
```

and corresponding 3 lines of output:

```
10  0 -10   0
11 12  13 100
21 12   3  36
```

Spreadsheets in this problem are rectangular. Each line has the same number of entries. Each entry is either an integer value or a formula. Entries are separated by one or more blanks or tab characters ('\t') in the input. There will be 0-10 rows and 1-26 columns. A specific entry is refered to by a letter (A-Z), indicating the column, followed by a digit 0-9 indicating the row. The top left cell is A0 and the bottom right is D2 in the example above.

Formulae begin with an '=' followed by one or more integer constants or cell references separated by + or - signs. There are no spaces within a formula.

One problem spreadsheet software usually has to deal with is circular references. E.g.:

```
=B0-3   =C0+3   =A0
```

Here A0 references B0 which references C0 which references A0. You may assume that such circular references do not occur in the input.

Note the spacing in the output. Numbers are separated by one or more spaces. There are no tabs in the output. The numbers in each column are printed so tha the least significant digits are aligned one above another. The spacing is suc that the longest number in each column is separated by 1 space from the previous column entry.

The sample input above is available in file a1.dat for testing. The corresponding output is in file a1.out. The judges will run your program using input redirection with other data.

Input is from System.in/stdin/cin and output to System.out/stdout/cout.

Problem b: Getting in Line

Computer networking requires that the computers in the network be linked. This problem considers a "linear" network in which the computers are chained together so that each is connected to exactly two others except for the two computers on the ends of the chain which are connected to only one other computer.

For various reasons it is desirable to minimize the length of cable used. Your problem is to determine how the computers should be connected into such a chain to minimize the total amount of cable needed. In the installation being constructed, the cabling will run beneath the floor, so the amount of cable used to join 2 adjacent computers on the network will be equal to the distance between the computers plus 4 additional meters of cable to connect from the floor to the computers and provide some slack for ease of installation.

Input

The input will consist of a series of data sets. Each data set occupies one line and contains the coordinates of the computers in a network.
Each network has at least 2 and at most 8 computers.
These coordinates will be integers in the range 0 to 150 meters.
No two computers are at identical locations and each computer will be listed once. The values are separated by 1 or more spaces.
Each computer location is represented by an x-value followed by a y-value.

Output

The output for each network should be the total length of cable required with distances in meters printed rounded to 2 decimal places.

File b1.dat contains three data sets:
```
0 1  0 2  1 0  1 3  2 0  2 3  3 1  3 2
1 0  0 1  1 1  2 1  1 2
0 1  1 0  2 1  1 2  1 1
```
The output for these data sets is 3 lines (available in file b1.out):
```
36.24
20.82
20.82
```
These numbers happen to be 32+3r, 18+2r and 18+2r where r*r = 2.
The second and third input data sets contain the same points. The answer in each case corresponds to connecting the points in the order shown in the second data set. An ordering for the first data set which gives the answer shown is 1 0 2 0 3 1 3 2 2 3 1 3 0 2 0 1.

Problem c: Cheap Fares

Consider the following 4 lines of input:

```
YYG YQM,AC,55 YHZ,AC,299 YHZ,CAR,270
YHZ YYT,AC,400
YQM YYT,AC,560 YHZ,CAR,102
YYZ YUL,TRAIN,200
```

and the corresponding 7 lines of output:

```
        YHZ     YQM     YUL     YYG     YYT     YYZ
YHZ     -       102     ?       182     400     ?
YQM     102     -       ?       55      527     ?
YUL     ?       ?       -       ?       ?       200
YYG     182     55      ?       -       607     ?
YYT     400     527     ?       607     -       ?
YYZ     ?       ?       200     ?       ?       -
```

The input gives the cost of travel between various destinations by different carriers. For each pair of destinations the output table shows the cheapest cost.

Each input line consists of a starting point followed by various "legs" separated by spaces. Each leg has the form destination,carrier,cost. There are no spaces except in front of each leg.

The third line of input above shows that you can travel from YQM to YYT on AC for $560 and from YQM to YHZ by CAR for $102. The first line of input shows that there are two ways to get from YYG to YHZ (on AC for $299 or by CAR for $270).
All "legs" can be used in either direction for the same cost.
For example, it costs $55 on AC to go from YYG to YQM. This means it is possible to go from YQM to YYG on AC and the cost would be the same.

In a long trip made up of several "legs" there is a $25 fee each time you change carriers. For example, the cheapest way from YYG to YYT is via AC to YQM, and then by CAR to YHZ and then on AC to YYT. Since there are 2 carrier changes the cost is 55+102+400+2*25=607. The $25 fee applies even to changes to CAR or TRAIN.
(If this problem is given as a class assignment it might be bestto not have a fee for changing carriers.)

The output lists all the destinations on the first line in lexicographic order with a tab character ('\t') in front of each one. The same destinations are repeated in the same order at the beginning of the following lines followed by the cheapest cost to get to each of the destinations. The cost of going to the same destination as the start point is shown as "-". If there is no way to get between the points using the given leg information the result is shown as "?".
All the costs are separated by a single tab character.

Input is from stdin/cin/System.in and output goes to stdout/cout/System.out.

Magic Decoder Ring

JBatman has lost his magic decoder ring, and he wants you to program a replacement. It turns out that the ring was not so magic after all, but simply employed a simple text encryption scheme of letter substitution. That is, the ring simply replaced each character in the text with another some distance from it in the alphabet, assuming a circular alphabet. The only trick was that the distance was different each time, and was determined from the received message. You see, each received message had exactly three occurrences of the word "batman" in it, at various positions. The ring simply determined which six letter word with the same pattern appeared exactly three times (and it could be the only one), and then figured out the distance.

For example, if the received text were (available as file d1.dat):

```
gsqqmwwmsriv ksvhsr geppmrk fexqer.
ksxleq gmxc riihw fexqer.
ksxleq qecsv omhrettih, liph ex ksxleq tevo.
gsqi uymgo fexqer, ksvhsr riihw csy!
wmkrih gsqqmwwmsriv ksvhsr.
```

The six letter words "ksvhsr", "fexqer", and "ksxleq" appear three times but the word "fexqer" must represent the word "batman", since "ksvhsr" and "ksxleq" do not have the correct pattern. Thus the distance used for the substitution is four (+4), since "b" to "f" is distance four. Thus to decipher the message, we move back in the alphabet four positions (-4) for each letter, giving: (file d1.out)

```
commissioner gordon calling batman.
gotham city needs batman.
gotham mayor kidnapped, held at gotham park.
come quick batman, gordon needs you!
Signed commissioner gordon.
```

Write a program that can decipher any text in this manner. Note that the text is written only in lowercase letters with punctuation, and that the punctuation is not encrypted. Also, the message texts tend to be short, with a maximum of 200 words.

Input is from stdin/cin/System.in and output to stdout/cout/System.out.

Problem e: Urban Elevations

An elevation of a collection of buildings is an orthogonal projection of the buildings onto a vertical plane. An external elevation of a city would show the skyline and the faces of the "visible"' buildings of the city as viewed from outside the city from a certain direction.

A southern elevation shows only the south faces of buildings and not the east or west sides. It shows the perfectly rectangular faces of buildings or parts of faces of buildings not obstructed on the south by other buildings.

For this problem, you must write a program that determines which buildings of a city are visible in a southern elevation, i.e. when viewed from far to the south.

For simplicity, assume all the buildings for the elevation are perfect rectangular solids, each with two sides that run directly east-west and two running directly north-south. Your program will find the buildings that appear in a southern elevation based on knowing the positions and heights of each city building.

Input

Consider the following 9 lines of sample input (available as file e1.dat):

```
0 7 2 1 3
1 8 4 1 2
0 5 5 2 4
3 2 3 3 3
4 1 1 1 2
5 1 2 1 2
5 0 1 1 1
0 4 3 1 3
1 0 1 4 2
```

Each line of input contains data for a single building
(5 float numbers separated by spaces in the following order):

    x-coordinate of the southwest corner
    y-coordinate of the southwest corner
    width of the building (length of the south side)
    depth of the building (length of the east/west sides)
    height of the building

Each map is oriented on a rectangular coordinate system so that the positive x-axis points east and the positive y-axis points north. Assume that all input corresponds to a legitimate map (no two buildings overlap when viewed from above).

The numbers above are all integer but in other data files they may be real. Buildings are numbered according to where their data lines appear in the map's input data building #1 corresponding to the first line of building data, building #2 data to the next line, and building #n to the nth line of building data for that map.

For this data the city viewed from above would appear as at the left below
where x = y = 0 is at the bottom left.

```
 2222   |
11      |
33333   |
33333   |
888444  |
 9 444  |       33333
 9 444  |       888444
 9  566 |       8984566
 9   7  |       8984576
```

The southern elevation for that city is illustrated in the diagram on the right.
Building 3 is taller than the others and obscurs buildings 1 and 2.

Output

The output for the input above would be one line (file e1.out):

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

This lists the buildings which are visible from the distant south. The output
is the numbers of the visible buildings as they appear in the southern elevation,
ordered west-to-east and south-to-north. This means that if building n and
building m are visible buildings and if the southwest corner of building n is
west of the southwest corner of building m, then number n is printed before
number m. If building n and building m have the same x-coordinate for their
southwest corners and if building n is south of building m, then the number n
is printed before the number m. For this program, a building is considered
visible whenever the part of its southern face that appears in the elevation
has strictly positive area.

Input is from stdin/cin/System.in and output goes to stdout/cout/System.out.

Problem f: Breaking Down Is Hard To Do

Given a coded text string s and a list C of n >= 1 alphanumeric codewords, determine if a valid parsing of s relative to C exists and, if so, output a smallest valid parsing of s relative to C. A parsing of s relative to C is valid if s is separated into k >= 1 codewords from C. Note that codewords may be repeated within a valid parsing. The smallest valid parsing is the valid parsing with the smallest value of k.

Your input will be an (n + 2)-line file in which the first line is the string s, the second line is the integer n, and the remaining n lines are the n codewords in C (one codeword per line) sorted in ascending order by codeword length.

Note: one input to this problem has a string of 67 characters and 13 codewords.

Sample input #1 (8 lines available in file f1.dat)

```
aababaaaa
6
a
b
aa
ba
aaa
aaba
```

Sample output #1 (1 line in file f1.out)

```
aaba + ba + aaa
```

Sample input #2

```
aababaaaa
6
a
b
aa
bab
aaa
ababa
```

Sample output #2

```
a + ababa + aaa
```

Sample input #3

```
aababaaaa
4
a
aa
aaa
aaba
```

Sample output #3

```
No solution
```

Input is from stdin/cin/System.in and output goes to stdout/cout/System.out.