University of Central Florida

2011 (Fall) "Practice" Local Programming Contest

Problems

Problem#	Filename	Problem Name
1	tnine	T9 Craziness
2	isograms	Pair Isograms
3	lawn	Lawn Maintenance
4	maybe	Ternary Logic
5	wacky	Waterford Wackiness
6	pickmen	Pick Men
7	ngram	Common Patterns
8	simpoly	Simple Convex/Concave Polygons
9	symm	Symmetric Diagonals

Call your program file:	filename.c, filename.cpp, or filename.java
Call your input file:	filename.in

For example, if you are solving Waterford Wackiness:

Call your program file:	wacky.c, wacky.cpp, or wacky.java
Call your input file:	wacky.in

UCF "Practice" Local Contest — August 27, 2011

T9 Craziness (filename: tnine)

Text-messaging is the current rage. Sitting in a boring class with nothing to do? Discreetly text your friend! The class isn't disturbed and you are no longer bored. It even allows you to make fun of that weird kid in class without actually saying a thing. But, as you may have noticed, regular texting, which includes pressing on each key up to four times to select the proper letter, gets tedious. Instead, you have discovered the T9 feature that allows you to press each corresponding numerical key once, but figures out exactly the word you meant to type, even though multiple letters correspond to each digit.

Here is exactly how T9 works: If you want to type a message to someone using a cell-phone pad, you only use the digits 2 through 9. The table below provides the corresponding letters for each digit:

Digit	Letters
2	a, b, c
3	d, e, f
4	g, h, i
5	j, k, l
6	m, n, o
7	p, q, r, s
8	t, u, v
9	w, x, y, z

If you wanted to type the message hello, you would just type 43556. Clearly there are many alphabetic strings that correspond to this five-digit string, but of all of those, hello is the only word. It turns out that for most words, no other word will correspond to the same exact set of digits. This is how T9 works – it cross references the digits pressed with a dictionary of words.

Obviously there are some instances where multiple words map to the same digits, such as "book" and "cool" which are both represented by "2665". In these cases, T9 selects the most frequently chosen word. (Phones with advanced T9 adapt to the user's typical word choices.)

For our purposes, your goal will be to write a program that translates messages typed in T9 to their alphabetic format. In the case that more than one message is possible, your program should simply calculate the total number of possible messages. (Note: This value should only be determined based upon the total number of possible combinations of words that could be formed and should not be restricted by the meaning of those words in any manner.) If no message is possible for the input T9 text, then your program should determine this as well.

The Problem:

Given a dictionary of valid words, and a set of messages sent in valid T9 format, determine the corresponding alphabetic message. If no message is possible, determine this. If multiple messages are possible, calculate how many are possible.

The Input:

The first line of the input file will contain a single integer n ($1 \leq n \leq 30000$), indicating the number of words in the dictionary. Each of the next n input lines will contain a single word from the dictionary. Each word will only consist of lowercase letters ($length \geq 1$) and there will be no duplicate words in the dictionary. Assume the input words start in column 1 and will not exceed column 80.

The next input line will contain a single integer, m, representing the number of messages to convert. Each of the next m lines will contain one message. A message will contain numeric strings (i.e., only digits 2 through 9) separated by spaces on a single line, with the first string starting in column 1 and each string being separated by a single space. No message will exceed 80 characters total, including spaces.

The Output:

For each input message, print a heading. Then, output one of the following three options:

If there is NO corresponding message, then print the following message out:

```
not a valid text
```

If there is exactly one possible message, then print out the corresponding message, all in lower case.

If there is more than one possible message, then print a message of the following format:

```
there are X possible messages
```

where X represents the total possible messages that could have been texted. You are guaranteed that X is less than $2^{31} - 1$.

Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

Sample Input:

```
11
hello
good
i
a
went
to
the
beach
cool
read
book
3
4 9368 86 843 23224
4 7323 2 2665 2665
8447 47 843 5278 8378 2273
```

Sample Output:

```
Message #1: i went to the beach

Message #2: there are 4 possible messages

Message #3: not a valid text
```

UCF "Practice" Local Contest — August 27, 2011

Pair Isograms (filename: isograms)

A word is considered to be "pair isograms" if each letter in the word appears exactly twice (not less, not more) in the word. For example, the word teammate is pair isograms since each letter in the word appears exactly twice (not less, not more), but the word dad is not since the letter "a" of the word doesn't appear twice.

Given a word, you are to determine whether or not it is pair isograms.

The Input:

The first input line will be a positive integer n, indicating the number of words to be processed. Each of the following n input lines contains a word, starting in column 1 and not exceeding column 52. Assume each input word will contain only lowercase letters (no other characters) and will be at least one letter.

The Output:

Print each input word along with a message indicating whether or not it is pair isograms. Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

Sample Input:

```
3
teammate
ali
dood
```

Sample Output:

```
teammate --- pair isograms

ali --- not pair isograms

dood --- pair isograms
```

UCF "Practice" Local Contest — August 27, 2011

Lawn Maintenance (filename: lawn)

George has been having trouble with his lawn. Not only does it seem to need fertilizing, but lately, he has found weeds popping up all over the place! Fortunately, the local home improvement store stocks a new version of a product, the improved Weed 'n Feed, that will fertilize grass and kill weeds at the same time. The product is sold in terms of 1000-square-feet of coverage. George doesn't know how much grass he has, so he's having a difficult time trying to figure out how much Weed 'n Feed to buy.

George does know a few things, though. George has the measurements of his lot, his house, and all of his landscaping beds that don't have grass. In order to figure out the area of his lawn, he needs to calculate the area of his lot, and subtract out the area of his house and landscaping beds.

Since George hates spending too much time/money, he's asked you to write a program that will help him figure out the optimum amount of fertilizer to buy. You'll have to calculate the area of his lawn, and then figure out how many bags of fertilizer to buy (to cover the area).

To make things simpler, the lot, house, and landscaping beds can all be described by simple (non self-intersecting) polygons (note that each polygon may be convex or concave).

The Input:

There will be multiple lawns to fertilize (George wants to share his program with the neighbors, too). Input for each lawn begins with an integer L ($3 \leq L \leq 10$), indicating the number of corners (vertices) on the lot. This is followed (on the same input line) by L pairs of integers, each pair specifying the X and Y coordinates of the position of each corner of the lot. The next input line for a data set will contain an integer H ($3 \leq H \leq 10$), indicating the number of corners (vertices) on the house. This is followed (on the same input line) by H pairs of integers, each pair specifying the X and Y coordinates of the position of each corner of the house. Following this will be an integer N on a line by itself, indicating the number of landscaping beds. Each of the next N lines will contain an integer B ($3 \leq B \leq 10$), indicating the number of corners (vertices) on that bed, followed by B pairs of integers, specifying the X and Y coordinates of the position of each corner.

All corners (vertices) will be specified consecutively in counter-clockwise order. Although vertices and edges may be shared, the house will be completely contained within the lot, and the landscaping beds will be completely contained within the lot and completely outside the house. Furthermore, the landscaping beds will not overlap each other, i.e., they won't have common area. You may also assume that the area of any polygon is always less than 2^{31}.

All coordinates will be separated by at least one space, and there may be leading spaces on each line (no trailing spaces). No characters other than digits, spaces, and newlines will appear in the input, i.e., no invalid input.

End of input is indicated by a value of zero for L (the number of vertices for a lot); this case should not be processed.

The Output:

For each lawn, print the number of fertilizer bags to buy. Use the following format:

```
Lawn #N: buy B bag(s)
```

where N is the number of the lawn (starting at 1), and B is the number of bags to buy. Note that the left-over fertilizer does not carry from one application (data set) to the next application. Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

Sample Input:

```
4   0   0  100   0    100 100     0 100
4  20  20   80  20     80  80    20  80
1
4  20  10   30  10     30  20    20  20
4  50  50  250  50    250 200    50 200
6 100 100  150 100    150 150   200 150   200 200   100 200
2
5  50  50   70  50     70  60    60  70    50  70
4 150 100  250 100    250 120   150 120
0
```

Sample Output:

```
Lawn #1: buy 7 bag(s)

Lawn #2: buy 21 bag(s)
```

UCF "Practice" Local Contest — August 27, 2011

Ternary Logic (filename: maybe)

Miles is an expert at Boolean logic, which uses the two values TRUE and FALSE and the fundamental operators NOT, AND, and OR. However, to understand his friend Maisy's way of thinking, he realized that he needs to understand her slightly more sophisticated form of logic, which has the additional value MAYBE. Maisy's ternary logic system has the same operators, and they yield the same results when applied to any combination of TRUE and FALSE values. So, Miles only needs to learn the following:

```
NOT operator:
     NOT MAYBE = MAYBE

AND operator:
     FALSE AND MAYBE = FALSE
     TRUE  AND MAYBE = MAYBE
     MAYBE AND MAYBE = MAYBE

OR operator:
     TRUE  OR MAYBE = TRUE
     FALSE OR MAYBE = MAYBE
     MAYBE OR MAYBE = MAYBE
```

Miles has noticed in some situations that Maisy says "maybe" quite often, and she seems to ignore conditions that Miles thinks should change her mind. In order to predict those situations, he develops expressions using her ternary logic, and starts to evaluate all possible outcomes of each expression in order to count the number of MAYBE results. Note that different results are possible for a given expression since any variable in the expression can take on the value TRUE, FALSE or MAYBE. Note, however, that when the same variable occurs more than once in the same expression, all copies of that variable take on the same value (i.e., all TRUE, all FALSE, all MAYBE).

The Problem:

Write a program to help Miles count the number of MAYBE results that are possible from a ternary logic expression.

The Input:

The first input line will contain a single integer n. The next n lines will each contain 1 to 40 characters which form a single logical expression. Each input expression will consist of the following characters only:

```
0 (zero) represents FALSE
1 (one) represents MAYBE
2 (two) represents TRUE
A, B, C, D, E, F, G (uppercase letters) are variables which can
     take on any ternary value
& (ampersand) represents AND
+ (plus) represents OR
! (exclamation) represents NOT, and always appears in front of
     what it operates on
```

Each expression is guaranteed to be well-formed. Precedence of operations is NOT, then AND, then OR. The associativity for each operator is left-to-right. There are no parentheses nor other grouping of sub-expressions within the expression.

The Output:

Print a heading for each input expression followed by the expression. Then print the number of MAYBE results that are possible from the expression. Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

Sample Input:

```
5
A&B
A+B+C
A&!B+B&!A
!A&A
2+A&!1
```

Sample Output:

```
Expression #1: A&B
3 MAYBE result(s) possible

Expression #2: A+B+C
7 MAYBE result(s) possible

Expression #3: A&!B+B&!A
5 MAYBE result(s) possible

Expression #4: !A&A
1 MAYBE result(s) possible

Expression #5: 2+A&!1
0 MAYBE result(s) possible
```

UCF "Practice" Local Contest — August 27, 2011

Waterford Wackiness (filename: wacky)

Orlando was recently ranked to be the "angriest" city in the United States by Men's Health magazine. One part of the ranking was based on traffic congestion and violations of the law. Most of you are probably aware of how poor the driving is within the Waterford Lakes Shopping Center. Four-way stop signs at intersections are seldom handled correctly with folks always going out of turn, etc. What is needed is a program to help show people the correct order that they should proceed through a four-way stop sign.

The Problem:

Given the times that cars arrive at a four-way stop sign, determine the order that they should proceed through the intersection.

The Input:

The first line will contain a single positive integer, n, representing the number of data sets. For each data set, there will be a line with an integer c ($1 \le c \le 100$), representing the number of car arrivals at the stop sign. Following this will be c lines, each containing a car entry. Each car entry will contain a positive integer, i, representing the number of the car, a single character, d (N, S, E, W), representing the cardinal direction from which the car is coming, and a positive integer, t, representing the time since the last car arrived from that same direction (or since the start of the current data set in the case of the first car in that direction). Each of the three elements of the car entry will be separated by a single space.

For each data set, assume that there will be only one entry for a given car number, i.e., a car arrives only once at the intersection. Also assume that no two cars will arrive at the intersection at the same time.

Note that the cars within a direction are listed in order (in the input). Also note that a car should proceed through the intersection as soon as it arrives (at the intersection).

The Output:

For each data set, output a header followed by the order of the cars as they proceeded through the intersection. For each car, output "Car #i" where i is the number of the car proceeding. Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

Sample Input:

```
3
2
1 N 30
2 S 40
3
1 N 30
2 N 20
3 S 40
6
3 E 400
4 W 100
1 N 300
2 S 200
6 N 150
5 W 50
```

Sample Output:

```
Data set #1:
Car #1
Car #2

Data set #2:
Car #1
Car #3
Car #2

Data set #3:
Car #4
Car #5
Car #2
Car #1
Car #3
Car #6
```

UCF "Practice" Local Contest — August 27, 2011

Pick Men (filename: pickmen)

Commander Oroojimar was piloting his spaceship carelessly one day when he struck an asteroid. His ship was badly damaged, forcing him to crash land on a nearby planet. Various bits of his ship were scattered across the surface of the planet. Luckily the planet was filled with friendly and helpful Pick Men who were eager to assist Commander Oroojimar in gathering the pieces of his ship. After gathering together a group of Pick Men, Commander Oroojimar, being particularly careless that day, led them into a dangerous cave. Now he needs your help to escape.

There are many different varieties of Pick Men. Each variety is impervious to a particular hazard, but susceptible to all others. Below is a table enumerating the different varieties of Pick Men and their immunities.

Pick Men Variety	Impervious to
Ruby	Fire
Topaz	Electricity
Sapphire	Water
Ivory	Poison

The cave Commander Oroojimar has led his Pick Men into consists of a number of chambers connected by passageways. The chambers are filled with deadly monsters. Unless Commander Oroojimar has enough Pick Men with him to defeat the vile fiends, he will be unable to survive passing through a chamber. Even if he has enough Pick Men to pass through a chamber, one Pick Man of each variety present (i.e., in Commander Oroojimar's possession) will be eaten by the monsters on the way through the chamber.

The passages connecting the chambers have their own dangers. They contain no monsters, but they do contain traps based on fire, electricity, water, and/or poison. If Commander Oroojimar has Pick Men with him that are impervious to the traps, they can be disarmed, allowing the passage to be safely traversed by all. Otherwise, the passage will be completely impassable.

The Problem:

Given the number of Pick Men Commander Oroojimar begins with and a description of the cave, determine the greatest number of Pick Men he can leave the cave with.

The Input:

Input begins with a positive integer indicating the number of data sets (caves) to be processed. The cave descriptions are on the following input lines.

The description of each cave begins with a line containing six integers. The first integer n ($2 \leq n \leq 50$) is the number of chambers in the cave (assume chambers are numbered 1 through n). The second integer m ($0 \leq m \leq 200$) is the number of passageways. The remaining four integers (all in the range of 0 and 500, inclusive) indicate the number of ruby, topaz, sapphire, and ivory Pick Men accompanying Commander Oroojimar into the cave, respectively.

Each of the next n input lines for a data set (cave) describes a chamber, with the i^{th} line describing chamber i. A description of a chamber consists of a single integer (in the range of 0 and 2000, inclusive) representing the number of Pick Men needed for Commander Oroojimar to survive that chamber. A value of 0 for a chamber indicates that there are no monsters and that the chamber can be visited without loss of Pick Men. Chamber 1 is where Commander Oroojimar enters, and chamber n is the exit, and both are guaranteed to contain no monsters.

The following m input lines for a data set (cave) describe passageways. Each line describing a passageway contains two integers followed by a string. The integers indicate the chamber number of the passageway's two endpoints (assume these two values are distinct and each between 1 and n inclusive). The string is either "N" to indicate no hazards, or contains one or more of the characters 'F', 'E', 'W', or 'P' (each letter appearing at most once) indicating the presence of fire, electricity, water, or poison, respectively. Assume that these input lines start in column 1 and there is exactly one space separating the values. Also assume that there will be at most one passageway between any two chambers.

The Output:

For each input cave, print a message on a line by itself in the following format:

```
Cave #i: message
```

where `i` is the cave number, beginning with cave #1, and message is either ``Commander Oroojimar can escape with j Pick Men.'', with `j` representing the maximum number of Pick Men with whom Commander Oroojimar can escape, or ``Commander Oroojimar is doomed.'' if he cannot escape the cave.

Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

Sample Input:

```
2
4 3 1 15 15 1
0
20
28
0
1 2 FP
2 3 EW
3 4 N
4 4 20 1 20 20
0
62
5
0
1 2 N
2 4 N
1 3 N
3 4 E
```

Sample Output:

```
Cave #1: Commander Oroojimar can escape with 26 Pick Men.

Cave #2: Commander Oroojimar is doomed.
```

UCF "Practice" Local Contest — August 27, 2011

Common Patterns (filename: ngram)

An "n-gram" is a contiguous sub-sequence of n items from a given sequence. For example, given the sequence "ALIGAME", its only 5-grams are ALIGA, LIGAM and IGAME. There are special names for the first few n-grams: 1-gram is called unigram, 2-gram is called bigram (digram), and 3-gram is called trigram.

You are to write a program that, given a paragraph, will find the most-frequently appearing unigram, bigram and trigram. We are interested in n-grams consisting of letters only. More specifically, you are to find the single letter that appears the most, the two consecutive letters that appear the most, and the three consecutive letters that appear the most. If there is more than one candidate for a given subsequence (e.g., several bigrams appearing the most), print the one that comes first alphabetically (i.e., smallest when compared as strings).

Note that "consecutive" letters means one letter immediately after another letter, i.e., no other characters (spaces or other separators) in between.

The Input:

There will be multiple data sets (paragraphs) to be processed. The first input line for a data set (paragraph) is an integer p ($1 \leq p \leq 50$) indicating the number of lines in the paragraph. The following p input lines provide the text (contents) for the paragraph. Each of these input lines will contain only lowercase letters, spaces, commas and periods. Assume that these input lines will not exceed column 70 and that each line will contain at least one letter. (Note that the only separators are spaces, commas, periods, and end-of-line.) End of data is indicated by a value of zero for p (number of lines for a paragraph).

The Output:

Print a heading for each paragraph, followed by its most-frequently appearing unigram, bigram and trigram (assume that each input paragraph will contain answers for each of these). Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

Note that for a string such as "aaaaaa", some interpretations view it as having three copies of "aa" and some view it as having five occurrences of "aa". Use the latter view for this problem (same concept applies to trigrams as well).

(Sample Input/Output on the next page)

Sample Input:

```
4
z z. z,z.  z z. z,z.
go  go  go
go  go  go
ali  ali   ali
3
a a. a,a.
bc    bc
abcd   abcd   abcd
0
```

Sample Output:

```
Paragraph #1:
   Unigram: z
   Bigram:  go
   Trigram: ali

Paragraph #2:
   Unigram: a
   Bigram:  bc
   Trigram: abc
```

UCF "Practice" Local Contest — August 27, 2011

Simple Convex/Concave Polygons (filename: simpoly)

Given two polygons, you are to determine their common area. Each polygon may be convex or concave, but each polygon will be simple (non-intersecting sides). Also, each side of a polygon will be parallel to x-axis or y-axis.

The Input:

There will be multiple data sets, each set consisting of three input lines. The first input line of a data set will contain two even integers m ($4 \leq m \leq 30$) and n ($4 \leq n \leq 30$), indicating the number of vertices for the two polygons, respectively. The next input line will have the x and y coordinates for the vertices of the first polygon. Similarly, the next input line will have the x and y coordinates for the vertices of the second polygon. Assume that input (x and y) values are integers between zero and 1000 (inclusive) and are separated by spaces. Also assume that the vertices for a polygon are given in order (i.e., the first two vertices for a polygon will give side 1 for that polygon and vertices 2 and 3 will give side 2, etc.). End of data is indicated by a value of zero for both m and n.

The Output:

Print a heading for each data set along with the common area. Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

Sample Input:

```
4 8
1 3     1 7     15 7     15 3
4 1     4 5     5 5     5 2     11 2     11 5     12 5     12 1
4 8
2 1     2 8     7 8     7 1
1 3     1 6     3 6     3 5     4 5     4 4     5 4     5 3
0 0
```

Sample Output:

```
Data set #1: 4

Data set #2: 6
```

The Output:

Print a heading for each input matrix. Then, echo print the matrix on consecutive output lines with a single space between letters. Then, for each request, print the message:

Symmetric diagonals r:

where r is the number of the symmetric diagonals requested. Print the symmetric diagonals on subsequent output lines. Print the upper diagonal before the lower, print the values as they appear from left to right in the matrix, and print a single space between letters. Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

UCF "Practice" Local Contest — August 27, 2011

Symmetric Diagonals (Filename: symm)

Consider the following matrix:

```
A B D F H
C A B D F
E C A B D
G E C A B
I G E C A
```

The elements labeled A are on the First *Symmetric Diagonal* of this matrix. The elements labeled B are on one of the Second Symmetric Diagonals, and the elements labeled C are on the other Second Symmetric Diagonal. Likewise, D's and E's are on the Third Symmetric Diagonals, and so on.

You are to write a program which, given a square matrix of characters, prints some requested symmetric diagonals.

The Input:

There will be several sets of input. Each input set will begin with an integer n $(1 \leq n \leq 15)$, indicating the size of the matrix. On each of the next n input lines, there will be n capital letters (with exactly a single space between letters) representing a row of the matrix. On the next input line will be a positive integer k, indicating the number of symmetric diagonals being requested from this matrix. On each of the next k input lines, there will be an integer representing a request. The integers representing requests are valid, i.e., they are guaranteed to be between 1 and n inclusive.
End of data is indicated by a value of zero for the matrix size (i.e., $n = 0$).

The Output:

Print a heading for each input matrix. Then, echo print the matrix on consecutive output lines with a single space between letters. Then, for each request, print the message:

Symmetric diagonals r:

where r is the number of the symmetric diagonals requested. Print the symmetric diagonals on subsequent output lines. Print the upper diagonal before the lower, print the values as they appear from left to right in the matrix, and print a single space between letters. Leave a blank line after the output for each data set. Follow the format illustrated in Sample Output.

(Sample Input/Output on the next page)

Sample Input:

```
2
T B
E O
1
1
4
F A S X
F R L O
L U E I
Q A N D
2
2
3
0
```

Sample Output:

```
Input matrix #1:
T B
E O
Symmetric diagonals 1:
T O

Input matrix #2:
F A S X
F R L O
L U E I
Q A N D
Symmetric diagonals 2:
A L I
F U N
Symmetric diagonals 3:
S O
L A
```