UTD Programming Contest for High School Students
October 28th, 2017

- **Time Allowed: three hours.**
- **Each team must use only one computer - one of UTD's in the main lab.**
- **Answer the questions in any order.**
- **Use only Java, C, or C++.**
- **Your program source code must be contained in one source file.**
- **Do not use the "package" construct in your Java code.**
- **Your programs must read from the named file specified in the problem header, and must output to System.out (cout for C/C++ programmers.)**
- **Do not access the web except to access Java C/C++, or Python documentation.**
- **Do not use any recording device containing code, other than UTD's computer.**
- **Your solutions must be entirely yours, typed by you during the time allowed for the contest.**
- **As soon as you have solved a problem, submit ONLY the source file via your `PC^2` client. Don't wait for the judges' response, keep going!**

Scoring

Equal points will be awarded to each question, even though they may not be equally difficult.

In order to break ties, we will use penalty points. The penalty points for a question will be zero if the question is not answered correctly. If a correct submission occurs for a question at time T minutes, then T penalty points will be added, plus 20 penalty points for each incorrect submission for that question.

The top ranked three teams, irrespective of category (Novice or Advanced) will be deemed 1st, 2nd and 3rd place-winners of the Advanced Contest. This rule in effect promotes any Novice teams to Advanced Status. Those teams cannot also be place-winners in the Novice Contest.

Of the remaining Novice teams, the top three ranked teams will be place-winners of the Novice category.

A. Sticks

Input File: A.txt
Runtime allowed = 10 seconds

George took several sticks of the same length and cut them randomly until each part became at most 50 units long. Now he wants to return sticks to their original lengths, but he forgot how many sticks he had originally and how long they were. Please help him. Write a program that computes the smallest possible original length of those sticks. All lengths are expressed integers greater than zero.

Input
The input file contains multiple test cases. Each test case is made up of two lines of text. The first line contains the number of stick parts after cutting. The second line contains the lengths of those parts separated by spaces. The last line of the file contains zero. Do not process this line.

There will be no more than 20 test cases and each test case will comprise no more than 12 stick parts.

Output
The output file contains the smallest possible length of original sticks, one per line.

Sample Input	Output for Sample Input
9 5 2 1 5 2 1 5 2 1 4 1 2 3 4 0	6 5

B. Digit Sum

Input File: B.txt
Runtime allowed = 10 seconds

Given two positive integers, X and Y, in radix r, $(2 \leq r \leq 16)$, X $\leq$ Y, sum the digits of all the numbers in the range [X,Y] and print the answer in radix r. The value of the answer will not exceed $2^{63} - 1$.

Input
The first line of the input will contain a single integer N, $(1 \leq N \leq 20)$ giving the number of ranges to follow.

N lines of text will follow, each containing radix r, expressed in decimal notation, followed by a single space character, and then the values of X and Y, expressed in radix r, and separated by a single space character. Digit values greater than 10 will be represented by upper-case alphabetical characters: A = 10, B = 11, . . ., F = 15.

Output
For each range, output a single line of text containing the sum of the digits within the given range. Digit values greater than 10 must be represented by upper-case alphabetical characters: A = 10, B = 11, . . ., F = 15.

Sample Input	Output for Sample Input
2 10 123 124 16 ABC AFC	13 7D5

C. Knight's Path

Input File: C.txt
Runtime allowed = 10 seconds

You are given a rectangular grid of n by m squares. One square contains a 'K' character, designating the starting place for the Knight, and one square contains a 'P' giving the position of the Princess whom the Knight wishes to court. Other squares are either occupied by '.' characters or '*' characters. Asterisks mark squares that the Knight must avoid. The Knight must make legal moves for a chess Knight while only visiting squares containing '.' characters on the grid, until he reaches the square marked 'P', where he and the Princess *may* live happily after.

Input:
The input will comprise multiple grids. The first line of input will be an integer N giving the number of grids to follow. Each grid will begin with a line containing two space-separated integers, R and C, giving the numbers of rows and columns of characters in the grid to follow. Then R lines of C characters follow giving the grid.

R and C will be in $[3, 20]$. N will be in $[1, 20]$

Output:
For each grid, output a single line of text containing the minimum number of legal moves that a chess Knight must make to move from its starting point to the square containing the princess. If the route is successful the Knight and the Princess will occupy the same square. If not, output the string "The Princess cannot be reached."

Sample Input	Output for Sample Input
2 3 4 .*P. K.*. .... 6 6 .*.... ...K.. .*...* ..*P*.	1 4

D. Land

Input File: D.txt
Runtime allowed = 10 seconds

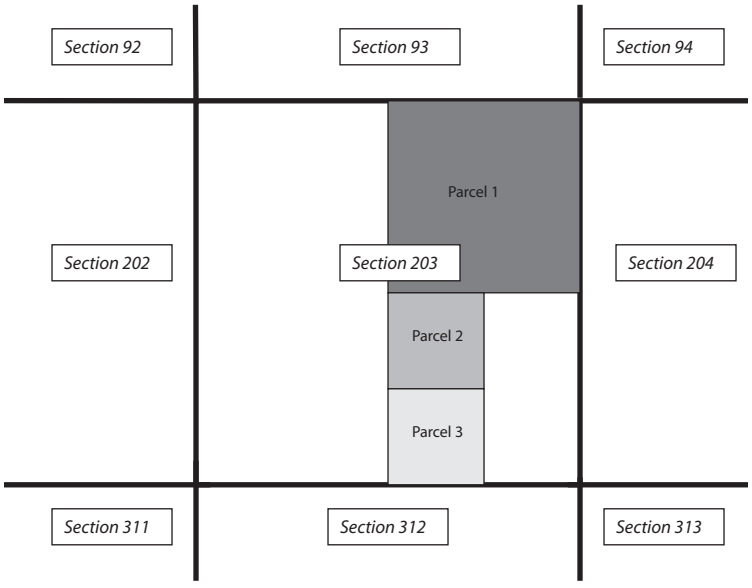
Julie has a received a bequest of some land. A list of her inheritance was as follows:

Parcel 1: Northeast Quarter of Section 203
Parcel 2: Northwest Quarter of Southeast Quarter of Section 203
Parcel 3: Southwest Quarter of Southeast Quarter of Section 203

She wondered what these terms meant. The Land Administration Website revealed the following:

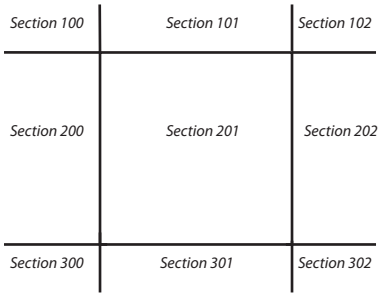
A section is a square-mile which equals 640 acres. Sections are numbered. You have to refer to the Land Administration Map to see the layout and geographic positions of the sections.
Sections may be subdivided into square-shaped quarters and those may be further subdivided into square-shaped quarters.
No parcel of land can be smaller than 1/16th of a section (40 acres).
Sub-sections are notated by North, South, East, West quarters of larger sections or sub-sections.
A rectangular parcel of land is listed as series of its square components.

Here is part of the Land Administration Map showing the area that includes Julie's inheritance:



How much land did Julie inherit? How many disjoint parcels are there? Two parcels are disjoint if they do not share any part of an edge. Two parcels meeting at a corner are disjoint.

Given a series of land parcels in this notation, calculate the total acreage and the number of disjoint parcels. To make the problem manageable, in all data sets, the Land Administration Map will be as follows:



Input:
The input file will begin with an integer T, $1 \leq T \leq 20$, on a line by itself giving the number of datasets to follow. Then T datasets follow, each one given in the following format.
A dataset begins with an integer P, $1 \leq P \leq 20$, giving the number of parcels of land in this dataset. P lines of text follow each one describing one parcel of land as illustrated in the Sample Input below.

Output:
For each dataset output one line of text containing two integers, the total acreage, and the number of disjoint parcels of land.

Sample Input	Output for Sample Input
2 3 Northeast quarter of Section 201 Northwest quarter of Southeast quarter of Section 201 Southwest quarter of Southeast quarter of Section 201 5 Northwest quarter of Northwest quarter of Section 101 Northeast quarter of Northeast quarter of Section 100 Northeast quarter of Northwest quarter of Section 101 Northeast quarter of Northeast quarter of Section 101 Northwest quarter of Northeast quarter of Section 101	240 1 200 1

E Polygons

Data File: E.txt
Time allowed = 10 seconds

A simple polygon is one in which the edges do not intersect each other and there are no holes in the polygon.

Given several simple polygons described by their vertices, calculate their areas.

Input:
The input begins with an integer T on a line by itself, giving the number of datasets to follow, $(1 \leq T \leq 20)$. Each dataset will begin with a line giving the number of vertices in the polygon, V, $(3 \leq V \leq 200)$. Then V lines follow, each giving a pair of space-separated integers, x y, representing the coordinates of a vertex, $(0 \leq x, y \leq 10^3)$. The coordinates will be given in counter-clockwise order around the polygon.

Output:
For each data set print a single line of text containing the area of the polygon rounded to, and printed with, one decimal place.

Sample Input	Output for Sample Input
2 4 0 0 10 0 10 10 0 10 9 0 10 6 1 11 1 14 3 14 6 12 8 8 8 8 6 4 10	100.0 74.0

F. Trips

Input File: F.txt
Runtime allowed = 10 seconds

> "How would you react if I said that I'm not from Guildford at all, but from a small planet somewhere in the vicinity of Betelgeuse?"
> Douglas Adams, The Hitchhiker's Guide to the Galaxy.

John and Jane met in a pristine resort hotel on a pristine remote island on the pristine planet of Vivalux. They love to travel and each has constructed a Bucket List of the places that they want to visit before retiring to an exclusive resort on a small planet somewhere in the vicinity of Betelgeuse.

John's list is:
Guilford, Earth; Aproprada Io; CozyKarma, Praxus; Manton, Mars; . . .

Jane's list is recorded similarly:
Guilford, Earth; CozyKarma, Praxus; London, Earth; Aproprada, Io; . . .

They need help in ordering their trips. To simplify matters the couple has agreed on a coding system for up to 26 places using lower case alphabetical characters.
Suppose Jane's list was: *qcaefgbhjv* and John's list was: *pckefbgrv*, implying that Jane wants to visit 'q' before 'c', and 'c' before 'a', and so on. John's list is also in the order that he wants to visit the places on his list. With these two bucket-lists they would select *cefgv* since it's the longest list that preserves the couple's individual ordering preferences.

Input:
The input will contain between 1 and 20 test cases (inclusive). The first line of input will be the number of test cases. Then each test case will follow comprising two lines of text:
The first line of each test case will be contain Jane's list of place symbols.
The second line will contain John's list of place symbols.

No bucket list will be more than 50 symbols long. There could be repeated symbols in each list.

Output:
For each test case, output the list of place-symbols created by the above process. There will only be one valid solution to each test case.

Sample Input	Output for Sample Input
2 acefgbhjv qcertfbgrv abcdgp aeqrefb	cefgv ab

G. Containers

Input File: G.txt
Runtime allowed = 10 seconds

A seaport container terminal stores large containers that are eventually loaded on seagoing ships for transport abroad. Containers coming to the terminal by road and rail are stacked at the terminal as they arrive.

Seagoing ships carry large numbers of containers. The time to load a ship depends in part on the locations of its containers. The loading time increases when the containers are not on the top of the stacks, but can be fetched only after removing other containers that are on top of them. The container terminal needs a plan for stacking containers in order to decrease loading time. The plan must allow each ship to be loaded by accessing only topmost containers on the stacks, and minimizing the total number of stacks needed.

For this problem, we know the order in which ships will arrive and the order in which containers arrive. Each ship is represented by a capital letter between A and Z (inclusive), and the ships will arrive in alphabetical order, A before B, B before C, etc. Each container is labeled with a capital letter representing the ship onto which it needs to be loaded. There is no limit on the number of containers that can be placed in a single stack.

Input:
There are between 1 and 20 (inclusive) datasets. Each problem consists of a single line containing a string of up to 50 upper-case alphabetic characters representing the order of arrival of a set of containers. For example, the line ABAC means consecutive containers arrive to be loaded onto ships A, B, A, and C, respectively. When all containers have arrived, the ships begin to arrive and are loaded in strictly increasing order: first ship A, then ship B, and so on. The input file ends with End of File.

Output:
For each problem give the minimum number of stacks needed to store the containers before loading starts.

Sample Input	Output for Sample Input
A	1
CBACBACBACBACBA	3
CCCCBBBBAAAA	1
ACMICPC	4

H Where's the Treasure?

Data File: H.txt
Time allowed = 10 seconds

The instructions on the treasure map were quite specific, that is, for a pirate:

> Three iron rods, A, B, C, stand proud on the beach. No living soul may move any one of 'em 'cos they spear through the 'arts of three king's mariners buried in those spots. The rods make a cursed triaaangle. The treasure is buried at a point I call T. That's what you'll need to find. No landlubber'll find it unless they're good at navigatin. If you was to draw a circle that passed through those three rods, it's center would be at T.

To be kind to landlubbers, in all test cases:

$$A_x = A_y = B_y = 0, \ (0 < B_x \leq 100.0) \text{ and } (0 < C_x, C_y \leq 100.0).$$

Input:
The input will contain from one to twenty datasets. Each dataset will comprise one line of text containing three real numbers, B_x, C_x, C_y

The input file ends with a line containing three zeros. Do not process this line.

Output:
For each dataset, output two numbers giving the x, y coordinates of the treasure, rounded to, and printed with, 4 decimal places.

Sample Input	Output for Sample Input
10.0 7.0 5.0 4.0 3.5 1.0 0.0 0.0 0.0	5.0000 0.4000 2.0000 -0.3750