

DKC3 2007 COMPUTING PROBLEMS (2)

1. Train Swapping

(75 pts)

At an old railway station, you may still encounter one of the last remaining "train swappers". A train swapper is an employee of the railroad, whose sole job it is to rearrange the carriages of trains. Once the carriages are arranged in the optimal order, all the train driver has to do is drop the carriages off, one by one, at the stations for which the load is meant. The title "train swapper" stems from the first person who performed this task, at a station close to a railway bridge. Instead of opening up vertically, the bridge rotated around a pillar in the center of the river. After rotating the bridge 90 degrees, boats could pass left or right. The first train swapper had discovered that the bridge could be operated with *at most two* carriages on it. By rotating the bridge 180 degrees, the carriages switched place, allowing him to rearrange the carriages (as a side effect, the carriages then faced the opposite directions, but train carriages can move either way, so who cares). Now that almost all train swappers have died out, the railway company would like to automate their operation. Part of the program to be developed is a routine which decides, for a given train, the least number of swaps of two adjacent carriages necessary to order the train. Your assignment is to create a routine that computes the minimal number of swaps. Each test case consists of two input lines. The first line of a test case contains an integer L , determining the length of the train ($0 < L \leq 50$). The second line of a test case contains a permutation of the numbers 1 through L , indicating the current order of the carriages. Each number will be separated with a space. The carriages should be ordered such that carriage 1 comes first, then 2, etc., with carriage L coming last. There will be 10 test cases, each separated by an asterisk. For each test case output the integer representing the minimal number of swaps to order the train.

Input File: C:\DKC3\TrainIn.txt

Output File: C:\DKC3\TrainOut.txt

Examples:

Input: 3
1 3 2
*

Output: 1

Input: 4
4 3 2 1
*

Output: 6

2. Curling**(75 pts)**

The sport of curling is played in ends. In an end, two teams throw eight rocks each toward a circular target. The centre of the target is called the tee. The team that scores is the team whose rock is closest to the tee. The score for that team is the number of rocks belonging to it that are closer to the tee than any other rock from the opposing team, and no more than six feet from the tee. Rocks with the same distance to the tee are considered to be tied. A rock is not considered to be closer to the tee than a rock with which it is tied. If the nearest rocks for the two teams are tied or both more than six feet from the tee, the end is declared to be a blank end, and there is no score. Assume that the tee is at the location whose coordinates are (0,0). The position of each rock is given as coordinate pairs (x,y) where x and y are in feet. For each end you are to determine which team wins, and its score. The first line of input contains the information for an end of curling for team A, the second line of input contains the information for an end of curling for team B. The first being the x coordinate separated by a comma and the second being the y coordinate. The coordinate pair will be separated by a space. There will be ten test cases and each test case will be separated with an asterisk. For each end, write a line containing the name of the team that wins (A or B) followed by a space followed by its score. If no team wins, write BLANK END.

Input File: C:\DKC3\CurlIn.txt**Output File:** C:\DKC3\CurlOut.txt

Examples:

Input: 2,3 2,2 3,3 4,4 5,5 6,6 7,7 8,8
 1,1 1,0 8,-8 -7,7 6,5 5,4 4,5 0,0
 *

Output: B 3

Input: -1,1 -2,-3 -1,-1 3,-6 2,-5 1,5 -3,3 -3,-2
 -4,5 3,2 2,-3 -1,-8 6,-3 -1,2 -3,0 4,0
 *

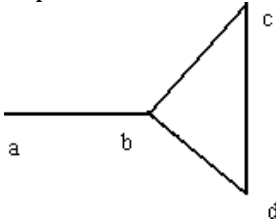
Output: A 2

3. The Forest for the Trees

(75 pts)

A graph G is a set of points $V(G)$, together with a set of edges $E(G)$, where each element of $E(G)$ is an unordered pair of distinct points of $V(G)$.

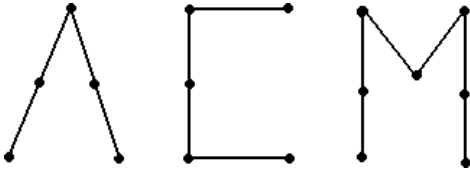
Example 1: Let G be a graph where $V(G) = \{a,b,c,d\}$ and $E(G) = \{(a,b),(b,c),(c,d),(d,b)\}$. The figure gives a depiction of G .



Notice that G contains the "cycle", $\{(b,c),(c,d),(d,b)\}$. A graph devoid of cycles is called a tree. A path in a graph G is an alternating sequence of points and edges, (beginning and ending with a point) such that all the points of the path are distinct. In the graph of example 1, $\{a,(a,b),b,(b,c),c,(c,d),d\}$ is a path. Every two points of a tree are joined by a unique path.

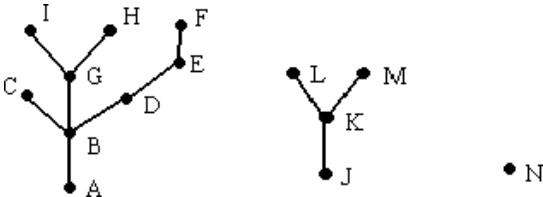
A graph is called connected if every pair of points is joined by a path. The graph of example 1 is connected. If a graph is not connected then it is made up of "subgraphs" which are. Each one of these subgraphs is called a connected component of the graph G .

A graph for which each connected component is a tree is called a forest, see figure below.



One extreme case worth mentioning is the case when one of the component trees has one point but no edges joined to it. This tree looks like an isolated dot. We will call this an acorn.

Example 2: Let G be a graph where $V(G) = \{A,B,C,D,E,F,G,H,I,J,K,L,M,N\}$ and $E(G) = \{(A,B), (B,C), (B,D), (D,E), (E,F),(B,G),(G,H),(G,I),(J,K),(K,L),(K,M)\}$. A depiction of this graph is given in the following figure.



Given a forest you are to write a program that counts the number of trees and acorns. There are ten test cases. Each test case will be followed by an asterisk (*) on a line by itself. Each test case is a forest description consisting of two parts:

1. A list of edges of the tree (one edge per line, given as an unordered pair of capital letters). If there are no edges (i.e. a forest without any trees), then a single dash (-) will be on the line.
2. A list of points of the tree. These will be given on one line with a maximum of 26 corresponding to the capital letters, A - Z.

For each test case your program should print the number of trees and the number of acorns, separated by a single space. For example: $x\ y$

Where x and y are the numbers of trees and acorns, respectively. Note x or y could be zero. This would be the case for a forest that has no trees or a forest that has no acorns. A forest may have no trees and all acorns, all trees and no acorns, or anything in-between, so keep your eyes open and don't miss the forest for the trees!

DKC3 2007 Computing Problems (2)

Input File: C:\DKC3\ForestIn.txt
Output File: C:\DKC3\ForestOut.txt

Examples:

Input: (A,B)
(B,C)
(B,D)
(D,E)
(E,F)
(B,G)
(G,H)
(G,I)
(J,K)
(K,L)
(K,M)
A,B,C,D,E,F,G,H,I,J,K,L,M,N
*

Output: 2 1

Input: (A,B)
(A,C)
(C,F)
A,B,C,D,F
*

Output: 1 1

4. Blood Type**(75 pts)**

Every person's blood has two markers called ABO alleles. Each of the markers is represented by one of three letters: A, B, or O. This gives six combinations of these alleles that a person can have, each of them resulting in a particular ABO blood type for that person (see chart 1). Likewise, every person has two alleles for the blood Rh factor, represented by the characters + and -. Someone who is "Rh positive" or "Rh+" has at least one + allele, but could have two. Someone who is "Rh negative" always has two - alleles (see chart 2).

Combination	ABO Blood Type
AA	A
AB	AB
AO	A
BB	B
BO	B
OO	O

Chart 1

Combination	Rh Factor
++, +-	+
--	-

Chart 2

The blood type of a person is a combination of ABO blood type and Rh factor. The blood type is written by suffixing the ABO blood type with the + or - representing the Rh factor. Examples include A+, AB-, and O-. Blood types are inherited: each biological parent donates one ABO allele (randomly chose from their two) and one Rh factor allele to their child. Therefore two ABO alleles and two Rh factor alleles of the parents determine the child's blood type. For example, if both parents of a child have blood type A-, then the child could have either type A- or O- blood. A child of parents with blood types A+ and B+ could have any blood type. In this problem, you will be given the blood type of either both parents or one parent and a child; you will then determine the (possibly empty) set of blood types that might characterize the child or the other parent. An uppercase letter "Oh" is used in this problem to denote blood types, not a digit (zero). Input will consist of 10 test cases. Each test case is on a single line in the format: the blood type of one parent, the blood type of the other parent, and finally the blood type of the child, except that the blood type of one parent or the child will be replaced by a question mark. To improve readability, white space may be included anywhere on the line except inside a single blood type specification. For each test case in the input print the set of possible blood types that is missing in the test case. If no blood type is possible, print "IMPOSSIBLE". If multiple blood types are possible, print all possible values in a comma separated list enclosed in curly braces {}. The order of the blood types inside the curly braces does not matter.

Input File: C:\DKC3\BloodIn.txt**Output File:** C:\DKC3\BloodOut.txt

Examples:

Input: O+ ? O-

Output: {A-, A+, B-, B+, O-, O+}

Input: AB- AB+ ?

Output: {A+, A-, B+, B-, AB+, AB-}

Input: AB+ ? O+

Output: IMPOSSIBLE