Computer Olympiad '98 Task Overview Day 1

All directories are located in 'C:\DAY1'

Note: Your programs are required to read input data from the current directory and write output data to the current directory. Do not use hard-coded paths

TASK:	Ladder	Flight	Animals
Directory:	LADDER	FLIGHT	ANIMAL
Prog. name:	LADDER.EXE	FLIGHT.EXE	ANIMAL.EXE
Input file:	LADDER.DAT	FLIGHT.DAT	ANIMAL.DAT
Output file	LADDER.OUT	FLIGHT.OUT	ANIMAL.OUT
Max time:	20 Seconds	20 Seconds	20 Seconds
Test cases:	5	5	5
Total points:	100	100	100

The Maximum score for day 1 is 300 points

Ladder

Word ladders

It is possible to arrange a list of words so that each word differs from the previous word by only one letter. The order of the letters may not be altered at any point. All words in the ladder have the same number of letters. A space is not a letter.

For example we can have a ladder from COAT to GOLD:

```
COAT
COAL
GOAL
GOAD
GOLD
```

A shorter version would be:

```
COAT
COLT
COLD
GOLD
```

Task:

You are given a dictionary of not more than 50 000 words. You are also given a starting word and an ending word. You are required to build the *shortest* word ladder, using only words from the dictionary provided, that transforms the starting word into the ending word.

Input:

The input will be present in two files, called LADDER.DAT and DICT.TXT. LADDER.DAT contains the starting word and the ending word, on different lines, in that order. Words will not be more than 6 characters long.

DICT.TXT contains the dictionary, with one word on each line. Words in the dictionary have various lengths, and consist only of characters in the ASCII table from ' A' to ' Z' (65 to 90). The words are sorted in the ASCII order, as indicated in the example. Any word appears only once in the dictionary.

Output:

A list of words forming the ladder in the file LADDER.DAT. Each word must be on a new line. The first word is the starting word, and the last word is the ending word. There will always be a solution.

Scoring:

Full marks are awarded for finding the shortest ladder.
A valid solution that is more than twice the length of the shortest ladder will score 20%.
A valid solution that is less than twice the length of the shortest ladder will score between 20% and 100%.
A solution with an invalid step in the ladder is awarded no marks.

Sample input 1:

DICT.TXT
A
AARDVARK
...
HARD
HAVE
HEART
...
SOFT
SORT
...
WARD
WORD
WORLD
WORT
...
ZULU

LADDER.DAT
HARD
SOFT

Sample output 1:

LADDER.OUT
HARD
WARD
WORD
WORT
SORT
SOFT

Sample input 2:

DICT.TXT
A
AMANZI
...
ILIZWE
ILIZWI
...
ISIZWE
IZIZWE
...

LADDER.DAT
ILIZWI
IZIZWE

Sample output 2:

LADDER.OUT
ILIZWI
ILIZWE
IZIZWE

Flight

African Air Flights

The African continent is served by many airlines with many airports. Much duplication of air routes takes place, and the Ministers of Transport met recently to see what could be done to reduce airline costs within the continent.

Your task is to identify sectors which may be eliminated, without inconveniencing passengers.

The Ministers were quite adamant that a passenger would only tolerate a certain number of sectors in a journey.

Definition: sector

A *sector* is a flight directly from one airport to another. For example, a flight from Tangiers to Kano is a sector. After takeoff at Tangiers, the aircraft will not land again until it reaches Kano. All sectors can be traversed in either direction – if there is a sector from Johannesburg to Cape Town, then you can assume that a Cape Town to Johannesburg sector also exists.

Definition: journey

A *journey* is a sequence of connected sectors.

An example of a 2-sector journey is Tangiers to Kano to Nairobi; an example of a 3-sector journey is Tangiers to Kano to Nairobi to Johannesburg.

Task:

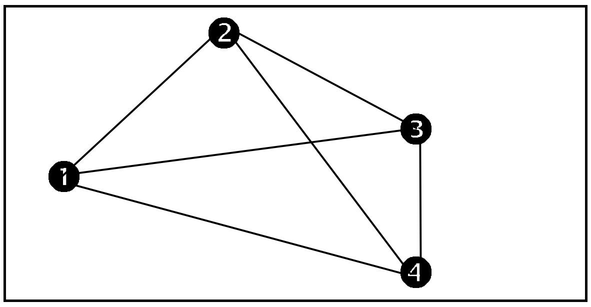
You are given the existing airport network which is a single connected graph, and all the sectors currently flown. You have to eliminate as many of the existing sectors as possible, while retaining the connectivity of the network as defined by the maximum number of sectors per journey. You may not add new sectors. The goal is to produce a network with the fewest sectors, which allows passengers to fly the same journeys as before.

Input:

The first line in the input file FLIGHT.DAT contains an integer N, giving the maximum number of sectors per journey. The next line is an integer S giving the number of sectors that follow. A list of sectors follows. Each sector is described by two integers, separated by a space. The integers represent cities, numbered sequentially from 1 to n where n is less than 1000. Each sector appears on its own line. Each city can only handle a maximum of 10 sectors.

Sample input:

FLIGHT.DAT	Explanation:
2	two sectors allowed
6	six sectors follow
1 2	sector between city 1 and city 2
3 1	sector between city 3 and city 1
4 1	sector between city 4 and city 1
2 3	sector between city 2 and city 3
2 4	sector between city 2 and city 4
3 4	sector between city 3 and city 4



Output:

The list of sectors which are members of the smallest network as defined in the task description. Each sector appears on its own line.

For example:

FLIGHT.OUT	Explanation:
1 2	sector between cities 1 and 2
1 3	sector between cities 1 and 3
1 4	sector between cities 1 and 4

Note: In this example there is more than one solution consisting of three sectors, but there are no solutions consisting of two sectors which allow a passenger to journey from any city (1, 2, 3, 4) to any other city.

Scoring:

Each data set will be scored as follows:

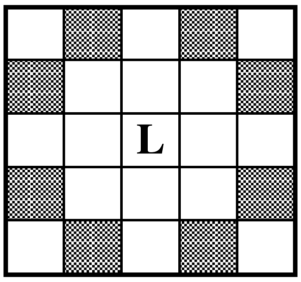
- If the solution is optimal, you will score 100%.
- If there is no improvement on the network given as input, you will score 0%.
- Otherwise you will score a percentage between 0 and 100, based on the number of sectors that you managed to eliminate as a percentage of the number of sectors that can be eliminated in the optimal case.
- If your output is invalid (eg. Containing new sectors, or requiring too many journeys for one route) you will score 0.

Animal

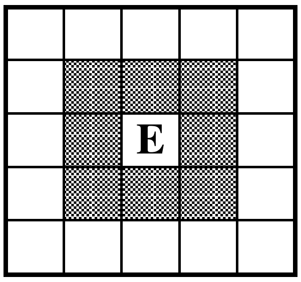
Invading animals

In a certain game played on a regular (8x8) chessboard, a number of animals are placed on the board. The animal can either be a *native* animal or an *invading* animal. The native animals will only move if the invading animal moves into its path, and then the native animal will kill the invading animal. A native animal dies if the invading animal moves onto it.

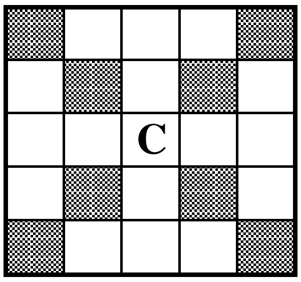
The different types of animals and their paths (shown on a 5x5 grid):



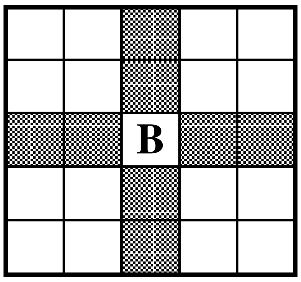
Lions
Move two squares horizontally or vertically and one square in a perpendicular direction.



Elephants
Move to any adjacent square.



Cheetahs
Move any number of squares diagonally but cannot move over an occupied square.



Bush pigs
Move any number of squares vertically or Horizontally but cannot move over an occupied square

Note:
Both invading and native animals consist of these four types. An animal cannot move over or through another animal, except the lion which jumps onto its target. There will be one invading animal on the board.

Task:

As input, you receive the positions of the animals on the board. Your objective is to create the shortest sequence of moves needed to kill all the native animals, without moving into their paths (which would result in being killed by a native animal).

Input:

The first value in the input file represents the total number of animals (N). The next N lines will be the X and Y position of each animal and a letter L,E,C,B representing the animal type. The last animal in the list is the invading animal, while the rest are all native animals. The square (1,1) represents the top left corner of the board. There will be at least 1 solution for each data set.

Sample input:

ANIMAL.DAT	Explanation:
3	3 animals
4 2 B	x = 4 y = 2 Bush Pig at (4, 2)
7 5 C	x = 7 y = 5 Cheetah at (7, 5)
1 1 L	x = 1 y = 1 Lion at (1, 1); invader

Output:

Write in the output file ANIMAL.OUT the sequence of moves the invading animal must follow to eliminate all native animals with the fewest moves. Express this as the co-ordinate pairs to which the invading animal moves, excluding the starting position.

Sample output:

ANIMAL.OUT	Explanation:
2 3	Moves to 2,3; eats nothing
3 5	Moves to 3,5; eats nothing
5 4	Moves to 5,4; eats nothing
7 5	Moves to 7,5; eats cheetah
6 3	Moves to 6,3; eats nothing
4 2	Moves to 4,2; eats bush pig

Scoring:

- An optimal solution will receive 100%.
- Regardless of previous correct moves, invalid moves receive 0% and further validation is halted.
- Sub-optimal solutions will score between 0 and 100%.
- A sub-optimal solution is one which requires more moves (by the invading animal) than the best known solution.
- 20% of the available marks will be deducted for each extra move made, in other words
- If your solution is 4 moves longer than the optimal solution, it will score 0 points.