DKC^3 2022 - Long Programming Problems

Important Reminders:

- Time remaining will be announced at 30 min., 5 min. and 1 min. left in the session.

- Assume leading zeros are used in all decimal notations unless the problem says otherwise. (ex: 0.167)

- Example input and output files are available in the **C:\DKC3** folder for each problem. These may be used to test your programs.

- Each program must read from its respective input file in the **C:\DKC3** folder and generate an output file in that same folder. The names of these files must match what is specified for each problem in this document. These output files will be retrieved electronically and scored by the judges at the end of the session.

- Do **NOT** store any code in the **C:\DKC3** folder. Anything stored there will be overwritten.

- At the end of the session, judges will overwrite the example input files with the official input files for each problem. Each file will contain 10 test cases. One member of each team will be asked by a competition organizer to run all completed programs. Programs may be recompiled at this point prior to being run, but no changes can be made to source code.

- Each program must complete execution within **two** minutes.

- After running your programs, be sure to close out of all IDEs.

DKC^3 2022 - Long Programming Problems

1. Missing Operator **(75 points)**

You will be given a set of two strings of digits separated by an equal sign (=). Write a program that will insert these operators (+, -, *, /, (,)) within the left string to make an accurate equation. Each of the operators may only be used once in an equation. The numbers formed may be more than a single digit.

Input
There will be one test case per line and 10 test cases total.

Output
The equation with the operators added should be output to the file.

Input File: **C:\DKC3\MissingIn.txt**
Output File: **C:\DKC3\MissingOut.txt**

Examples:
Input:
23=5
1265=90

Output:
2+3=5
(12+6)*5=90

DKC^3 2022 - Long Programming Problems

2. CrossCribb **(75 points)**

CrossCribb is a variant of the common card game, cribbage. CrossCribb is played with 4 players (2 teams of 2) with a standard deck of 52 cards and a 5x5 playing board consisting of 25 spaces for cards to be placed. Teammates sit across from each other around the playing board and take turns placing cards one at a time. The object of the game is to create the best-scoring cribbage hands in your 5 columns, while simultaneously minimizing your opponents' scoring opportunities in their rows.

Scoring:

- Any number of cards that add up to 15 (regardless of suit) – 2 points
    - Face cards are valued as ten for this purpose.
- Runs of three, four, or five cards – 3, 4 and 5 points respectively
    - Runs are considered by card rank: Ace (1) through 10; then Jack, Queen, King
    - Aces are considered "low" and runs cannot "wrap around" – Q-K-A is not a run.
    - Multiple runs are possible: a hand containing 2-2-3-4 is two runs of three cards.
    - You must take the maximum possible run. A hand containing 2-3-4-5-6 is a run of five and cannot be downgraded to multiple runs of four and three.
- Two, three, or four of a kind – 2, 6 and 12 points respectively
- Flushes: 4 or 5 cards of the same suit – 4 and 5 points respectively

Note: cards can be used more than once for each combo. A five, a queen, a jack, and a king, all of spades, count as three fifteens (5 and Q, 5 and J, 5 and K) and a four-card flush.

Input
Your program should take in a completed CrossCribb board with 5 lines of input, one for each row on the board. Each line will consist of 5 cards separated by a comma. Each card will be designated by a rank: Ace through 10 or Jack (11) – Queen (12) – King (13) as well as a suit: Hearts, Clubs, Spades and Diamonds. There are 10 test cases, each separated by an asterisk (*).

Output
Your program should calculate the scores for the team scoring ACROSS (left to right), and the team scoring DOWN (top to bottom). Output should consist of the final score of the game separated by a hyphen, followed by a comma and a space, then the winning team (either ACROSS or DOWN). The higher score should always be listed first in the output. In the event of a tie, output the final score followed by a comma and a space, then the word "TIE".

Input File: **C:\DKC3\CrossCribbIn.txt**
Output File: **C:\DKC3\CrossCribOut.txt**

Examples:
Input:

JS,3S,KC,7S,6S
4C,KS,7D,9H,KD
4H,10D,QH,10H,QS
AD,8H,9S,2D,4S
QD,10C,8D,2S,9C
*
10D,6C,5D,7S,2D
3S,7H,8H,QH,AH
10H,10C,2S,QC,9S
6S,8D,3H,5C,QD
KH,QS,JS,KD,AS
*

Output:
21-17, DOWN
23-21, ACROSS

DKC^3 2022 - Long Programming Problems

3. Secret Bases (75 points)

A certain shadowy organization maintains many secret bases throughout the city, but they keep having to move them when they are discovered. They want to hold their monthly meetings at the most centrally located secret base, but since they are always moving they have to recalculate which base that is every month, and that takes way too much effort. The members also don't like to travel straightforward routes that they could easily be spotted on, so they have to take into account the travel times between various bases, and that there isn't a good way to travel directly between some of the bases.

Given a list of bases, their locations, and the paths between them, your job is to determine which base is the fastest to reach, on average, from every other base. It is guaranteed that every base will be reachable from every other base by following some sequence of paths. If two bases are equally fast to get to on average, select the one that appears first alphabetically.

Input
The first line of each test case will consist of a list of secret base names, separated by a comma and a space. There may be up to 100 total bases. The second line will contain an integer N, and the next N lines will denote paths as two consecutive base names and an integer describing the time it takes to travel that path. All paths are traversable in both directions. There will be one empty line between test cases.

Output
The output for each test case should be a single string – the name of the secret base for which average travel time from each of the other bases is minimized.

Input File: C:\DKC3\SecretBasesIn.txt
Output File: C:\DKC3\SecretBasesOut.txt

Examples:
Input:
North, South, West, Central, East
6
North West 6
Central West 2
East North 4
South Central 3
South East 3
South North 11

Redbase, Greenbase, Bluebase, Orangebase, Yellowbase, Whitebase
8

Redbase Orangebase 15
Orangebase Yellowbase 24
Yellowbase Whitebase 12
Whitebase Bluebase 15
Bluebase Greenbase 16
Greenbase Redbase 20
Bluebase Orangebase 50
Orangebase Greenbase 30

Output:
South
Bluebase

DKC^3 2022 - Long Programming Problems

4. Bloxorz Solver **(75 points)**

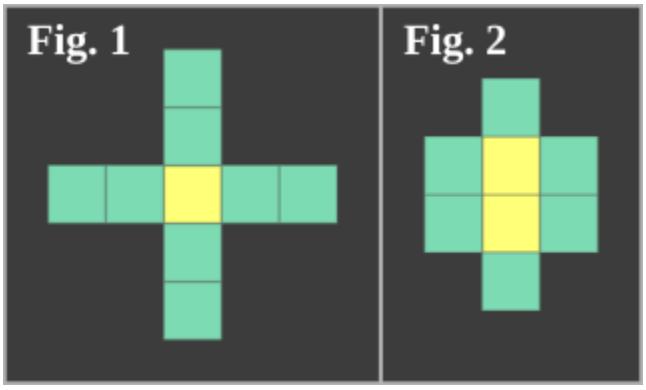
This problem is inspired by Bloxorz, a flash-based 3D puzzle game where the objective is to maneuver a block to have it fall through a square hole.

Objective
Your goal is to maneuver a rectangular cuboid with dimensions 1 x 1 x 2 on a 2-dimensional grid made up of 1 x 1 square tiles. While moving the cuboid around, your block must never, at any point, have any part of its bottom-facing surface exposed to open air.

Input
You will receive an array of strings describing the layout of the grid to be traversed. A 1 represents solid ground (tiles), a 0 represents open air, a B represents your starting position, and an X represents the square hole (destination).

Output
You must output a string representing the minimum sequence of moves required to get the block into the square hole. It will consist of a combination of the following characters: U (up), D (down), L (left), R (right). If there is more than one possible solution, use the highest route on the grid (top left to bottom right).

Block Movement



The block can cover either one or two tiles with each movement, depending on whether it is standing upright or on its long end. The image above shows an overhead view; the yellow squares represent the tiles occupied by the block and the green squares represent the tiles occupied when moved toward a given cardinal direction.
In Fig. 1, the block's position is standing upright, and in Fig. 2, the block is on its long end.

Technical Details
Input will always be valid and there will always be a solution.
The block always begins in an upright position.

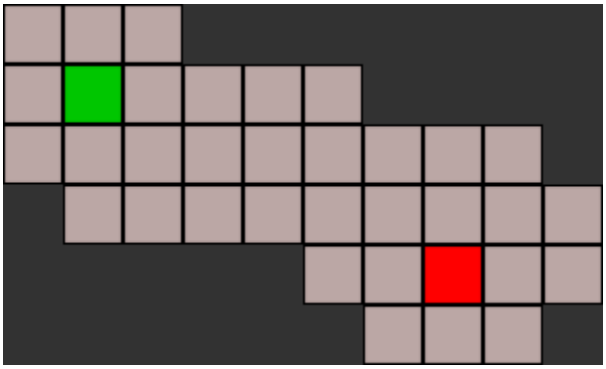
The destination exit does not count as open air.
The maximum grid size will be 15 x 20 (rows x columns)

Input File: **C:\DKC3\BloxorzSolverIn.txt**
Output File: **C:\DKC3\BloxorzSolverOut.txt**

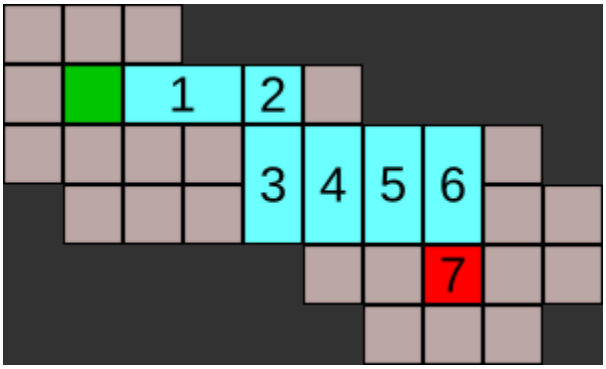
Examples:
Input:
1110000000
1B11110000
1111111110
0111111111
0000011X11
0000001110
*
000000111111100
111100111001100
111111111001111
1B11000000011X1
111100000001111
000000000000111

Output:
RRDRRRD
RURRRULDRUURRRDDDRU

An overhead view of the game grid for the example is shown below, with a green square representing the starting position of the block, the red square representing the square hole, the light grey squares representing the platform tiles, and the dark grey areas representing open air:



Below is the sequence of moves to solve this map, numbered and highlighted in blue.


1
2
3
4
5
6
7