Problems for OpComp 2025

By Dr. Robert W. Hasker

Friday 21 November 2025

Notes

- Unless a problem specifies otherwise, you may assume that all user inputs are valid.
- The precise wording of your output is not critical as long as it is completely clear and meets any stated requirements.
- There is generally no runtime limit for solutions, but if a solution runs far longer than deemed reasonable it will be terminated and marked as incorrect.

See the OpComp website for additional instructions and rules.

1. My Pet

ASCII art is a time-honored way to capture images. Write a program that prints an ASCII art picture (that is, using the keyboard characters) of a pet or favorite zoo animal. For example,

```
             /\--/\
             | oo |
            `+----+'
             |    |_
            /       \
           /| |   | |
          / | |   | |
|      ___| | |--| |
\     / ____\_/  \_/
 \==='-/
```

If you cannot do better, we will be sad, but you will still get full credit. Your image must be at least 6 lines tall and 20 characters wide and must look approximately like an animal. There is no extra credit for beautiful images! Remember that `\` is a special character in most programs and that it may need to be doubled (`\\`) to print correctly.

2. Attack of the Clones

This problem is based on problem 1, but you can solve them in either order. Write a program that reads a number from the user and displays that many copies of your pet picture, horizontally, with a consistent number of spaces (1 to 4) between each clone. For example, if the input was 3, the cat in the first problem would be tripled:

```
             /\--/\              /\--/\              /\--/\
             | oo |              | oo |              | oo |
            `+----+'            `+----+'            `+----+'
             |    |_              |    |_              |    |_
            /       \           /       \           /       \
           /| |   | |          /| |   | |          /| |   | |
          / | |   | |         / | |   | |         / | |   | |
|      ___| | |--| |  |      ___| | |--| |  |      ___| | |--| |
\     / ____\_/  \_/  \     / ____\_/  \_/  \     / ____\_/  \_/
 \==='-/               \==='-/               \==='-/
```

With the personality of some cats, a scary thought. You can assume the entered value is at least 1, and see the first problem for a discussion of special characters like `\`.

3. Telephone

In the popular telephone game, people get in line, and someone whispers a message to the first. That person then turns and whispers what they thought they heard to the second person. This repeats, with the last person announcing the message they thought they heard. Hilarity ensues. For a simple analysis of the game, we can assume there is a percentage chance that a listener will hear and speak each word correctly. Say it is 80%. Then a three-word phrase would have 51.2% chance of making it to the second person in the line since 0.8^3 is 0.512, and about a 26.2% chance of it being transmitted correctly to the third person since 0.512^2 is approximately 0.262. The program is to print the number of people that can be in the line before the chance of transmitting a message correctly falls below 25%.

Write a program to read two lines of input. The first line is a sentence, and the second is a (possibly fractional) transmission rate. Words in the sentence are separated by a single space, and any punctuation is right after a word. No input will be more than 50 words.

```
Live, laugh, love!
80
```

Output the number of people that can be in the line that would have a at least a 25% chance of being transmitted correctly. In this case, that would be 3.

4. Cave Mice

Your neighbor has given you an ASCII art image of mice in a cave. Write a program to count the mice, represented as o's. The following image has ten. The floor of the cave is always marked by a line of just hash marks, and this is the only place where hash marks will appear in the input. Each line of input will be less than 100 characters.

```
                __          _________
               /  \      /           \
 .-----   ----/    |    /             \
/      \_/         \___/o     o       \
|o             /\              o/\     |
|_o_/\_o__o/\/  \_____________/  \_ooo|
####################################
```

No mice are outside the cave, but they can appear anywhere inside it. The above example is available on the contest site as `4.txt`.

5. AI Stutterer

An AI system to transcribe spoken words into written text is overly sensitive and records a large number of occurrences of stuttering even when the person speaking rarely stutters. The stutters that need to be fixed have transcribed the same letter three or more times in a row. Upper and lower case are treated as the same. This is not a common pattern among real people, and it clearly needs to be fixed. You have been asked to count the number of stutters recorded in a transcript so this data can be used to refine the transcription algorithms. Read a paragraph of input, defined as text ended by a blank line, and report the number of times the system recorded a stutter. Note that very long stutters (say, 10 repetitions of a letter) are counted as single occurrences. The input always ends with a blank line, and there are no other blank lines in the input. Each stutter must be in a word, so stutters cannot cross word separators such as blanks, punctuation, and line endings.

A large sample input is available on the problem website as `5.txt`.

6. Four Square

Write a program to find four-letter words in squares such as

```
c a
t s
```

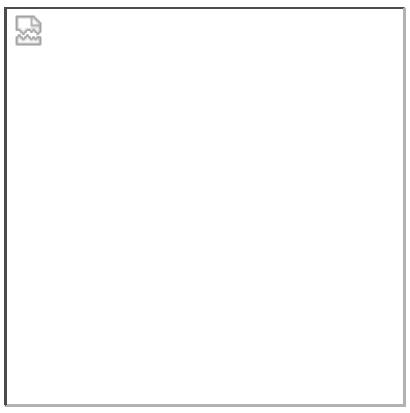
All four letters must be used, each must be used just once, and once you have traveled one diagonal then you cannot travel the other. For example, the above square contains acts, cast, and cats but not scat. Hint: these rules mean there is a relatively small number of possible words to consider.

Using the provided dictionary, write a program to read a list of squares and write the unique, matching words for each square in alphabetical order. The first line of input is the number of squares, and each square is given as two lines. For each square, print its number followed by an alphabetical list of the *unique* words that match all four letters in each square. If there are no matches for a given square, write "no matches".

7. Sailing the Sparkling Sea

Serenity & Sparkle Sailaway rents motorized yachts and would like your help in planning tours for groups of four clients. Clients select reefs for snorkeling, and your program is to print a list of directions. Reefs are mapped on a grid, and you must translate those to headings that clients will follow. The yachts travel at 6 knots, where one knot is equivalent to traveling 1852 meters (a nautical mile) in one hour. The yachts start at an island in the center of the map, and coordinates of are given as kilometers east and kilometers north. Locations to the west or south have negative distances (that is, going away from the. east or north). The yachts have simple automation keeping them on a particular heading, so there is no need to account for wind and currents.

Your program is to read a list of way points, each given as kilometers east followed by kilometers north from the starting point, one per line. Some way points have a location name (which may be multiple words) after the two numbers. There is no punctuation between the fields; they are separated by spaces. The location names are for reefs. For each way point, write the heading and time needed to travel to that location. Headings are expressed in the traditional way, as degrees from north, such as shown in the following compass:



Times are expressed in cruise minutes to reach the way point. If the way point represents a reef, the group is given three hours to snorkel and otherwise explore the location. They then set out for the next way point. The process repeats until they return to the dock location where the yacht started, 0 east and 0 north.

In addition, compute the total drinkable water needed for the tour, assuming each person in the group drinks 4 liters per day. Since the water is delivered in 25-liter containers, display the number of containers needed for the full group for the planned voyage.

For output, list the following for every way point:

- If the way point is a reef, display the reef name; otherwise, write “unnamed way point”
- Display the coordinates (north, east)
- Display the required heading
- Display the travel time At the end of the trip, display its total duration in hours (including snorkeling time) and the number of 25-liter water containers that are required. The format of this information is not important, but display any floating-point values with a single digit of precision. You can display additional information, but not so much as to make the output difficult to read.

Clarification: North is 0 or 360 degrees, East is 90, South is 180, and West is 270.

8. Letter Box

In an earlier problem, the goal was to find four-letter words in a two-by-two grid. This problem extends it to larger grids such as

```
d a e
n g r
```

In particular, the grid is always two rows with each row having the same number of letters. For this problem, the rules are

- All letters must be used in a solution exactly once.
- One letter follows another if it is in any one of the 8 adjacent positions (left, right, up, down, or on the diagonals).
- Diagonals cannot cross; for example, if g follows a in a word, then n cannot follow d. This means you can follow a diagonal only if all of the other letters on the “back” side of the diagonal have already been used.

The above grid includes danger but *not* garden or gander.

Write a program to read a grid in the form of a two-dimensional array of letters and display all words from a dictionary that solve that grid. All words must be in the provided dictionary. Grids will be limited to 16 letters maximum (that is, two rows of eight). The first line of input will be the number of columns. Print the matching words in alphabetical order, or print ‘no solutions’ if there are none.

9. Tree Travel

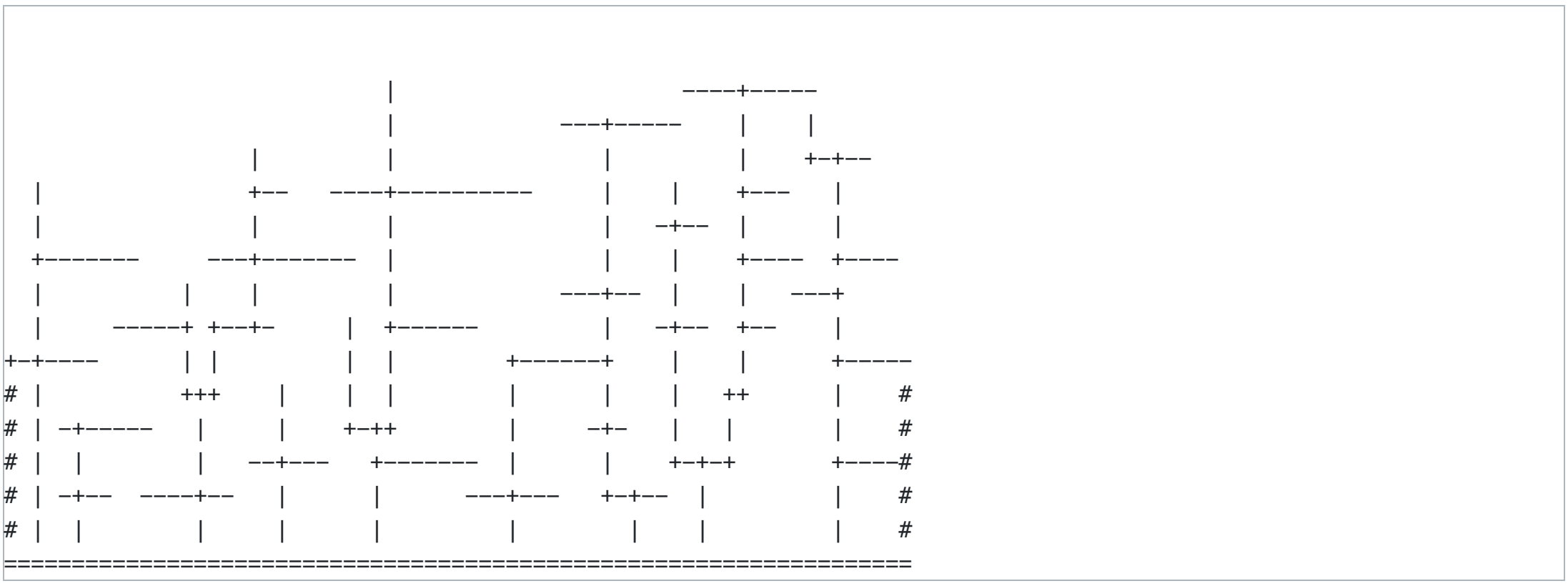
Tarzana is fleeing from robots through a forest. She can climb any tree that can hold her weight and she can jump across small gaps from one branch to another. She can drop up to 12 feet without being injured. The robots cannot climb, so she is fine in the trees, but cannot touch the ground. She is currently in the western camp and has learned how to disable the robots, but she needs to get this

information to the eastern camp to use it. The robots are poor at predicting where she will go, but good at remembering where she has been, so her overall direction of travel must be east to west, meaning she never walks westward on a branch (but if the trunk leans west, she can still go up it).

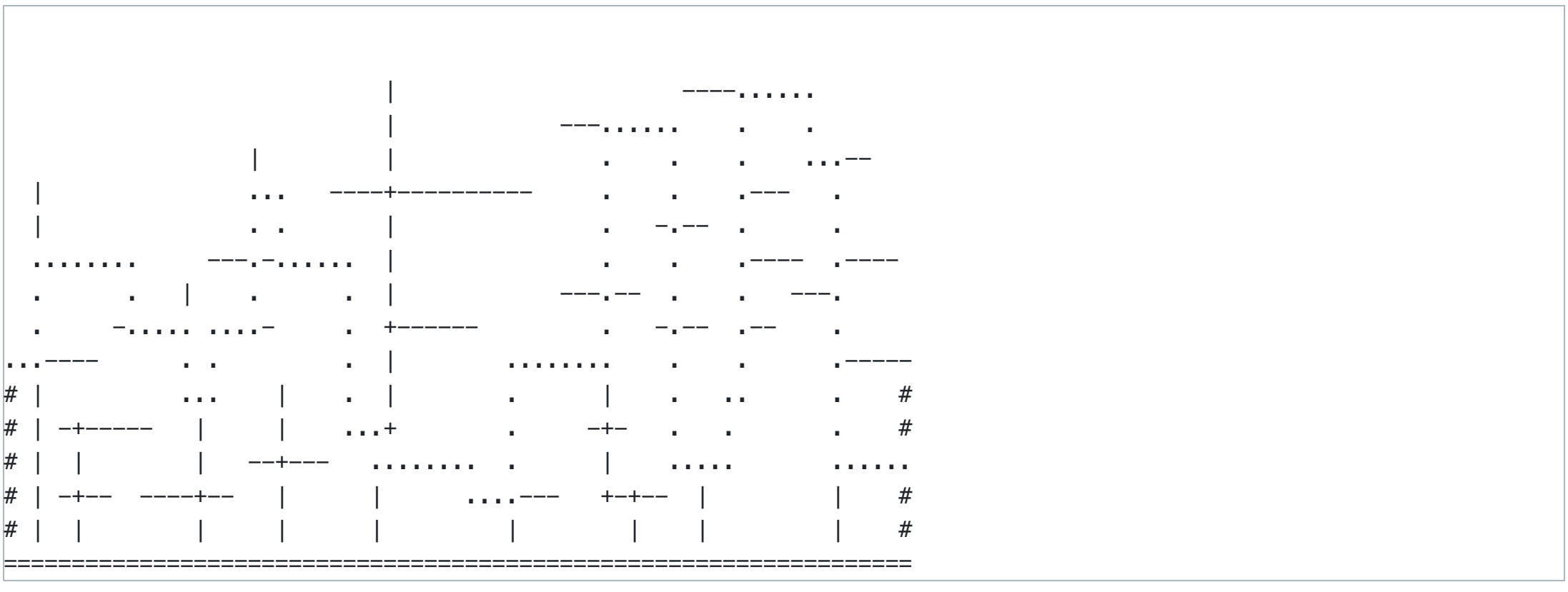
Tarzana has a schematic of the forest that she will use to plan her route. The western and eastern camp walls are marked by `#`, `|` marks trunks Tarzana can climb, `-` marks branches she can walk on, and `+` marks branching points. She can climb any direction from a branching point (up, down, sideways) as long as there is support in that direction. The ground is marked by `=`. The western camp wall is at the far left, the eastern camp wall at the far right. The walls are not climbable. Each character in the schematic represents a region of space that is 6 feet tall and 3 feet wide. Tarzana can go up or down trunks and she can walk to the right (east) on branches. When she reaches the tip of a branch, she can drop to another trunk, branch, or wall segment directly below the tip, but only if that drop does not cross more than 12 feet. She cannot drop from other places on branches because they are too slippery. There are places where she can climb from one tree to another or from the wall onto a tree, those places are also marked with `+`.

Read the schematic (noting you have reached the end when you read a `=`), and write it back out with periods (`.`) showing Tarzana's path. If there is more than one path, you can show any valid solution.

For example, if the schematic is



then her path could be



There are other paths that are also valid. If there is no valid path, write a message stating so. The schematic is never more than 20 lines, and the width of each line is less than 100. The start point is always in column 0, 6 characters above the forest floor. The end point is always in the same row but the rightmost column.

Hint: whenever there is a choice, add the alternatives to a list of locations to explore. Whenever a dead end is found, take the last alternative from the list and attempt to explore it. You may also find it useful to keep track of how you reached each node so you can reconstruct the path later. It may also be helpful to write a function that can determine all possible ways to move from any given position.

The above example is available on the contest website as `9.txt`.

10. Influencing the World

Social media sites make it easy to determine who knows who. A friend has invented a new “must have” product. The details are classified, but it is similar to hula hoops, hacky sacks, fidget spinners, and Tamagotchi. You have obtain a data file with information about who follows whom on social media sites. The idea is to send the product to just enough people to ensure that if each of those people post about the product, everyone will hear of it. (You tried to explain to your friend that not everyone will post about a random new product, but your friend is confident that if they also give people some cash to post about the product, they will do so.)

Your friend has asked you for a program that reads a file containing follower information and then display the *minimum number* of sample products that must be given out so that all followers will hear of the product. The input is a number of lines, and each line contains an user’s name followed by the names of those users who follow that first user. Note that if Dick follows Jane, Jane does not necessarily follow Dick. For example, if the input is

```
6
Ann12 Doug9
Brenda18 Zoe5
Charles2
Doug9 Charles2
Xander2
Zoe5 Brenda18
```

then the output would be

```
3
```

since Doug9 follows Ann12 and Charles2 follows Doug9, Brenda18 and Zoe5 follow each other, and Xander2 follows no one. Case is not significant in names, and names have just letters and digits (no spaces). The above input is available on the contest site as `10.txt`.