Problems for Op 2022

By Dr. Robert W. Hasker
with the assistance of the MSOE Competitive Programming Students and Dr. James Lembke
Friday 18 November 2022

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. There is generally no time limit on solutions, but if a solution runs far longer than deemed reasonable it will be terminated and marked as incorrect.

1. It's Cold Out (10 points)

To many people, a day feels especially cold when it is much colder than the previous day's temperature. A swing from 60 (Fahrenheit) to 40 in the fall will feel cold, while a temperature of 35 can feel normal if the previous days have averaged from 30 to 38. Write a program to read a list of high temperatures for a series of days and print which days have a drop of 15 degrees or more. The first number will be the number of days. For example, if the input is

8 59 60 45 50 51 45 40 19

then the program is to print which day numbers have a difference of 15 degrees or more and the temperature difference found. For example, the above input would result in printing

Temperature drop (15 degrees) on day 3
Temperature drop (21 degrees) on day 8

Note that the first temperature listed (in this case, 59) is for day 1. If no pairs have a temperature drop of 15 or more degrees, print "No cold days". If it simplifies your solution, you can assume all input is on a single line and the first number accurately counts the number of temperatures to follow.

2. Word in a Word (10 points)

A simple vocabulary game is to write a long word on the board and have students identify words that fits in it. For example, if the word is 'algorithm', then words that can be found in it include 'hit', 'tail', and 'grill'. Write a program that reads a series of words and prints the words which are invalid. For example, if the input is

deciduous dice tide us deci deed dire duo

then the output would be

Invalid words: tide dire

All input will be lower case letters with the words on a single line and separated by spaces. Note that letters can be repeated. For this problem, a word can be in itself; both 'tar' and 'rat' are "in" 'tar'. There is no requirement that non-words be marked as invalid.

3. Magic Number (10 points)

In Major League Baseball (MLB), teams play 162 games over the course of the season with the hopes of winning their division and advancing to the playoffs. To win their division, a team must have the best win-loss record; that is, the most wins and consequently the fewest losses. As the season progresses, each of the trailing teams lose enough games that even winning all remaining games would be insufficient to win the division. This leads to the calculation of a "magic number" that represents the combination of wins needed by a trailing team and losses by the division leader that are needed for that trailing team to reach first place. The magic number for each trailing team in the division is computed by the formula

$$M_T = T_G - W_L - L_T + 1$$

where

- M_T = the magic number for a trailing team,
- T_G = the total number of games played in a season,
- W_L = the total wins by the leading team, and
- L_T = the total losses by the trailing team.

If a team's magic number is zero or negative, then that team is eliminated from playoff contention (that is, that team cannot achieve first place). The following table shows magic numbers for a division:

Team	Record		Magic Number
	Wins	Losses	
Detroit Tigers	88	56	-
Minnesota Twins	75	71	4
Cleveland Guardians	73	70	5
Kansas City Royals	76	68	7
Chicago White Sox	65	78	eliminated from playoff contention

The input to the program is n lines of text where n is the number of teams in a division. As the MLB has 30 teams in it, n is between 2 and 30. Each line will be a team's wins and losses, with the teams listed by their order in the standings. The end of the input is marked with a pair of zeros. The output is to be the magic number of each trailing team, again in the order of standing. If a team has been eliminated, write "eliminated from playoff contention" rather than printing a number.

For example, if the input is

```
88 56
75 71
73 70
76 68
65 78
0 0
```

then the output is

```
4
5
7
eliminated from playoff contention
```

Note multiple teams can be listed as being eliminated.

4. Crossing Bridges (20 points)

You find that you are a test subject in an experiment conducted by an oddly named company, Aperture Science. Many of the tests include crossing bridges to traverse escape rooms. In addition to spanning hazardous areas, the bridges have steps that go up and down multiple levels. You can step up or down one level, and you have special boots that allow you to fall multiple levels without harm, but you cannot go up more than one level at a time. You have also found a map under a slice of cake; this map gives schematics of the bridges so you can determine which are crossable. You will use this information to plan your escape routes.

To speed up the search for a route, you have decided to write a program to evaluate bridge schematics. The input is a sequence of named bridges, and the output is the name of each bridge followed by "OK" or "NOT CROSSABLE" indicating whether the bridge is crossable or not. More specifically, the input file is a series of records where the first line of each is the name (on a full line of text; names can include spaces) followed by a series of lines indicating the levels spanned by the bridge. Bridges can have any length from one segment to 40 segments. Each segment is marked with an 'x'. If there is no segment, then you would fall through to any segment below or (if there is none) to the center of the planet. Bridges can have up to 40 levels. The bottom-most level will either be followed by the end-of-file or by a plus sign ('+'). If the end is marked by a plus sign, then you are guaranteed there will be another complete bridge schematic record.

The input is a side view of the bridge with an 'x' marking bridge segments and spaces marking missing segments. For example, the input schematic could be

```
x xx
 x       xxxx
        x
     xxx
```

In this case the output would be "OK" since you would fall from to the 'xx' at the top to the 'xxx' at the bottom but be able to work your back to the 'xxxx' and escape. The provided data file `input4a.txt` contains some other examples and `output4a.txt` contains the corresponding output. You can assume lines will not have trailing spaces.

5. Sally Sells Seashells (20 points)

Sally sells seashells by the seashore and business is booming. Every day, Sally receives orders for shells of different sizes, and she then takes just enough buckets to the beach to carry all of the shells. Each bucket can hold up to (and including) 100 grams of shells. She wants to carry just enough buckets to the beach, so she is writing a program to determine the minimum number of buckets needed to fill the orders. For example, if the input is

```
50
100
25
```

(meaning one 50-gram shell, one 100-gram shell, and one 25-gram shell), then the output is

```
2
```

since two buckets are sufficient. In general, the program reads a series of integers representing the grams for each shell in the order, and it prints the minimum number of buckets needed to transport that order to Sally's seashore-side shop. Each shell weight is on its own line, and the program is to read weights until it reaches the end of the input. You can assume no order has more than 100 shells.

6. Worst Words (20 points)

Wordle is a popular game on the internet in which the player has six tries to guess at a five-letter word. Each guess must be a legal, 5-letter word, and the program shows which letters are correct. For example, a player might have the sequence

S T ***A*** I R
A ***S*** P ***E*** N
L A P S E

In this example, the underlined letters are in the correct position and the bold italicized letters are correct but in the wrong position. A particularly challenging problem is when the player has three letters in the correct position but a large number of alternatives for the remaining two positions. For example

? A C ? O

(which has correct letters in positions 2, 3, and 5) could be the words WACKO, MACHO, and CACAO. A player can quickly get to the point where there are more alternative words to fill out the unknowns than there are guesses left in the game. Your program is to read a list of words and identify the largest group of words which all have three letters in common. The common letters must be in the same locations in each word in the group. The program is to print the words in the group in alphabetical order. The input consists of up to 100 five-letter words, all in upper case with one word to a line. The output will be the list of words with the greatest number of 2-letter misses. If there is more than one group with the same number of words, print the one with the alphabetically smallest first word.

```
BRINE
CHASE
CRAZE
DRIVE
KHAKI
PRIDE
PRIZE
SPARE
STAIN
```

then the output is

```
BRINE
DRIVE
PRIZE
PRIDE
```

since all have RI and E in the same position. Another candidate is CHASE and CRAZE (with C, A, and E at common positions), but this group has just two words in it.

7. Wave After Wave (40 points)

While walking the seashore near her home in South Carolina, Sally notices how waves leave lines on the beach. After her success with computing how many buckets she needed to fill orders, she wonders if she could write a

program to determine whether each wave line "fits" within the line left by the wave before it - that is, whether one wave line completely fits within another. She plans to use this data to train a neural net to predict tides for the stretch of seashore she searches. This stretch runs north/south with the ocean to the east, so you can assume all waves hit the shore from the east, north-east, or south-east.

More specifically, the input is a series of waves lines represented by coordinate pairs x,y where the y axis parallels the shore. The coordinate system is positioned (with x = 0) such that the y axis is at the average high water mark for the year. The coordinates are listed in order from north to south, one to a line, so if x_1,y_1 is followed by x_2,y_2, then you know $y_1 > y_2$ and that segment of wave followed a straight line from point (x_1,y_1) to point (x_2,y_2). All coordinate values are integers, though you will likely need to use floats for calculations. The data for each pair of waves is separated by a line containing just an asterisk:

```
*
```

Note there will be no asterisk after the last wave. The first wave is numbered 1, the second 2, and so on. The coordinates of each wave in a file will have the same maximum and minimum y values. The program is to print the numbers of (adjacent) waves which fit within each other. For example, the input

```
3,10
4,5
2,3
3,1
*
2,10
6,4
1,1
*
4,10
7,5
8,1
```

would result in printing

```
wave 3 fits in 2
```

There can be more than one pair of waves that fit in an input. If all waves cross, print

```
no waves fit
```

Note that the units used for the coordinate system do not affect results. You can assume every wave has at least two pairs of coordinates.

8. Speedy Simon's Seashell Service (40 points)

Sally adds a new product line, painted seashells. These become popular, so she hires her friend Simon to deliver painted seashells to clients who cannot come to her store at the sea shore. Sally is writing a program to plan routes for Simon to follow when delivering packages, and she has contracted you to estimate how long it will take to travel between different points on the routes. The input to your program is a number followed by that many locations. These locations are on the route from the starting point to the ending point. Each location is on its own line using the format

name distance1 transportation1 destination1 distance2 transportation2 destination2 …

That is the line starts with a location name and then lists the destinations that can be reached from that location along with the distance (in miles, a floating-point value) and the transportation options for getting there. All locations (the starting point and the destinations) are single, alphanumeric words, and all other data on the line is separated by single spaces. The transportation options are single letters:

- b: city bus
- g: greyhound bus
- c: car
- p: plane
- t: train

One other option, walking, is *always* available. For example,

MadisonCampus 73 c Rockford 0.5 cb MadisonDepot 5.5 cbt DaneCountyAirport 201 c Peoria 270 c Springfield
MadisonDepot 72 cg Rockford 6.5 bt DaneCountyAirport 202 gct Peoria 265 cg Springfield
DaneCountyAirport 79 cgp Rockford 210 cgp Peoria 275 c Springfield
Rockford 136.0 cg Peoria 212 cgt Springfield
Peoria 73.5 gctp Springfield
Springfield

This shows that from the center of the UW-Madison campus, you can take a car for 73 miles to Rockford; a car or bus for 1.5 miles to MadisonDepot; and a car, bus, or train for 5.5 miles to DaneCountyAirport. Similarly for the other locations. Springfield is the destination on this trip, so it has no routes to anywhere else. Note that not all pairs of locations have transportation options, and sometimes the options are counterintuitive. For example, if there's a bus from A to B and then from B to C, you would think there's a bus from A to C. But that is not always the case; Sally's system filters out options that will not work because of issues such as arriving too late to catch the last bus going in the right direction and routes that would backtrack to a previous location. You can assume that if a location is mentioned in the input, there is a descriptive line for that location. Locations will not be in any particular order except that the first location is the starting point and the last one is the final goal. This data is available as `input8a.txt`.

Different modes of transportation have two variables: the average speed and the average wait once while waiting for the transportation alternative to become available. This is defined by the following table:

Method	Average Speed (mph)	Average Wait Time (minutes)
city bus	25	10
car	50	15
greyhound	65	30
plane	325	90
train	45	30
walking	3	0

This captures such issues as needing to stop less for breaks on a greyhound bus, having to slow trains down as they go through towns, and needing to be at the airport early to get through security.

Your task is to write a program which computes the fastest time (in minutes) to arrive at a destination. For example, getting from MadisonCampus to Springfield (to the nearest minute) takes 248 minutes. This is accomplished by taking a car to DaneCountyAirport, flying to Peoria, then taking the greyhound to Springfield.

Your program is to read a list of locations from the input (until there are no more locations available) and find the quickest route from the first location listed to the last location listed. The input always has at least two lines. The output is the number of minutes rounded to the one's place. For example, a more accurate calculation of the time to reach Springfield is 248.2 minutes, but this is printed as 248o. If there is no way to get from the start to the final goal, print

```
no route
```

9. **Tiled Floor** (40 points)

Sam recently hired a tile installer to tile a kitchen floor. The floor was supposed to be very colorful with no two adjacent tiles having the same color. Unfortunately, the installer failed to ensure adjacent tiles have different colors. The mortar is still wet, but it is difficult to lift just one tile at a time, so the installer is limited to swapping adjacent tiles. The question is how to exchange adjacent tiles in as few moves as possible so that the floor meets the criteria that no two adjacent tiles have the same color.

A sample input is

RGR
RPC
GRB
YPG

Representing the tiles on a three by four floor where *R* is a red tile, *G* is green, *B* is blue, *C* is cyan, *P* is purple, and *Y* is yellow. In general, the input will be a series of lines of R, G, B, C, P, and Y. Each line will have the same length and there will be a maximum of 15 lines with 15 characters in each line. The output is to be an integer representing the number of swaps of adjacent tiles. For this problem, "adjacent" means touching and in the same row or column; for example, the only two tiles are adjacent to the yellow tile in the lower left corner of the above floor: the green tile at the start of the third row and the purple tile in the middle of the fourth row.

The output for the above floor will be

2

since the red tile at the start of row 2 can be swapped with the green tile at the start of row three, and then the red tile in middle of row 3 can be swapped with the blue tile at the end. This gives the arrangement

RGR
GPC
RBR
YPG

Other fixes are possible such as exchanging the first two tiles on row 2 to get PRC and then exchanging the middle tiles in rows 3 and 4. Your program does not print the resulting floor arrangement, just the minimum number of tile swaps that must take place. Sometimes it is not possible to fix a floor:

GGYGP
CGGRG

This isn't possible to tile because there are 6 Gs and a floor this size can fit only 5 without two being adjacent. In cases where there is no solution, the output is to be

`not possible`