COMPUTER SCIENCES

George Fox University H.S. Programming Contest Division – II April 10, 2021

General Notes

1. Do the problems in any order you like. They do not have to be done in order
(hint: the easiest problem may not be the first problem)

2. Scoring: The team who solves the most problems in the least amount of time with the least submissions wins. Each wrong submission will receive a 20 min time penalty that will only be added to the time score once the problem has been successfully solved. Time is calculated for each problem as the total time from the start of the contest to the time it was solved.

3. There is no extraneous input. All input is exactly as specified in the problem. Integer inputs will not have leading zeros.

4. Your program should not print extraneous output. Do not welcome the user. Do not prompt for input. Follow the form exactly as given in the problem.
(hint: spaces? No spaces? What does spec say!)

5. All solutions must be a single source code file.

A. Bowling

You are a bowler and want to create a program to calculate bowling scores. The program will read in values for each frame then print out all 10 scored frames.

In frames (1 to 9) the bowler will roll once or twice, trying to knock down all 10 pins. When all the pins are knocked down in the first throw the frame will store an 'X' and the bowler will not throw a second ball. When pins are still standing after the first throw, the bowler will throw a second time.

When 0 pins are knocked down in a throw a '-' is added to the frame. When a bowler knocks down some of the standing pins in a throw, the number of pins knocked down by that throw is added to the frame. When all the pins are knocked down by the second throw a '/' is added to that frame.

Examples:

Frame Data	First Throw	Second Throw
--	0 pins	0 pins
X	10 pins	
6-	6 pins	0 pins
-5	0 pins	5 pins
-/	0 pins	10 pins
4/	4 pins	6 pins

The final frame in bowling is different than the rest. If a bowler gets a strike on their first throw or a spare on their second throw, they get a total of three throws in the final frame. After each throw, if there are no pins standing, all the pins are stood back up for the next throw. Also strikes and spares in frame do not receive bonus points for future throws.

Examples

Frame Data	First Throw	Second Throw	Third Throw
--	0 pins	0 pins	0 pins
X--	10 pins	0 pins	0 pins
6-	6 pins	0 pins	
-/5	0 pins	10 pins	5 pints
XXX	10 pins	10 pins	10 pins
X6/	10 pins	6 pins	4 pins
X-8	10 pins	0 pins	8 pins

The score of each frame will be the previous frame's score plus the current frame's value. The value of the current frame is the number of pins knocked down in the frame plus any bonuses to the frame. When a spare is made the frame's value is increased by the number of pins knocked down on the next throw. When a strike is made the frame's value is increased by the number of pins knocked down on the next two throws.

(continued on next page)

Input
The first input will be an integer N indicating the number of complete games to follow. Each game will appear on its own line and each frame will be separated by a comma.

Output
Display the frame scores of each game on its own line with commas separating each frame.

Example Input:
```
3
-8,-/,X,X,8-,--,8/,8/,-/,X-8
X,X,X,X,X,X,X,X,X,XXX
-/,5/,6/,X,X,8/,-5,54,5/,XXX
```

Output to screen:
```
8,28,56,74,82,82,100,110,130,148
30,60,90,120,150,180,210,240,270,300
15,31,51,79,99,109,114,123,143,173
```

B. Cargo

A shipping company has contracted you to write a program that will maximize their profit. They have a single ship can only carry up to 100,000 tons of cargo. They want the program to analyze offers and determine which offers will net them the most profit.

Each offer will include the company's name for the contract, how much they will pay to transport their goods and the number of tons they need shipped. In order to take a contract all of its cargo must be taken.

Input
The first input will be an integer N indicating the number of contracts to evaluate. Each contract will be in the following format:

```
companyName-tonsOfCargo/payment
```

Output
Display the names of the companies for the taken contracts, in alphabetical order each on their own line. Followed by a line that displaces the tons on the ship and total of the payments, in the following format:

```
### tons - $####.##
```

Example Input:

```
5
Bobcat-51200/13020.65
Sopi-30000/8077.56
Vocal-20000/4000.44
Babel-20000/4000.87
Tort-25000/6451.25
```

Output to screen:

```
Babel
Bobcat
Tort
96200 tons - $23472.77
```

This page intentionally left blank

C. Voltage

Your physics teacher asked you to write a program that will solve Voltage problems, using the formula V=IR (V-Voltage, I-Current, R-Resistance). The program will take in 2 of the 3 values and solve for the final value.

Units:

- (V) Voltages – volts
- (R) Resistance – ohms
- (I) Current - amps

Input

The first input will be an integer n indicating how many data sets will be given. The format for each data set will be:

```
     value units, value units
```

Output

Display the missing value with 2 decimals places and its units, in the following format:

```
     ##.## units
```

Example Input:

```
4
8 volts, 9 amperes
4 amperes, 3 volts
6.8 amperes, 4 ohms
4.2 volts, 1.2 ohms
```

Output to screen:

```
0.89 ohms
0.75 ohms
27.20 volts
3.50 amperes
```

This page intentionally left blank

D. Cave Rescue

You are building a program for a rescue team that saves people trapped in caves. When the team goes into a cave, they will have a map of the cave system. The map will be including walls, passages, flooded passages, exits and where the trapped person is. When the team reaches a trapped person, they will have to decide if they can be taken out safely or if they should remain in place until conditions change.

When a survivor is reached, they are asked if they are able to swim or not. People that cannot swim cannot travel through flooded passages. People that can swim can be moved through up to 3 flooded passage at a time. People that cannot be safely moved are supplied food and water.

Key:

- P – Passage
- F – Flooded Passage
- E – Exit
- T – Trapped Person
- W – Cave Wall

Note: Some caves may have more than 1 exit.

Input
The first line will contain a single integer n that indicates the number of scenarios that follow. The first line of each scenario will contain whether the person can swim, the number of columns in the cave and the number of rows in the cave. The format for this data will be:

```
canSwim,numberOfColumns,numberOfRows
```

Following this input will be the cave map.

Output
Display “Rescue” if the person can be rescued or “Supply” if they cannot.

(continued on next page)

Example Input:
```
5
false,2,2
EF
FT
true,4,3
EPWW
FFFW
WWTW
true,4,4
EWWE
FFFP
FFFF
WWWT
true,5,5
FEWWW
FFPPP
FFPWP
FPPPP
TWWWW
true,5,5
WWWEW
FFFFE
PWWWW
FPPPP
TWWWW
```

Output to screen:
```
Supply
Rescue
Rescue
Rescue
Supply
```

E. Cricket

In the dart game Cricket, players can score points by hitting 15, 16, 17, 18, 19, 20, or a bullseye.

Before a player can score on a number, they must "lock out" that number by hitting it a total of 3 times. After a player has locked out a number, they will score that number of points per hit (on that number) if their opponent has not yet locked out that number. Hitting a number will not grant points if both players have locked that number out.

The numerical values can be hit for a single, double, or triple. A single counts as 1 hit, a double counts as 2 hits, and a triple counts as 3 hits. The bullseye does not need to be locked out. 1 hit on the bullseye is 25 points and 2 hits on the bullseye is 50 points. The bullseye can only receive up to 2 hits per throw.

EXAMPLE: Adam hits 20 twice. Bella hits 20 twice. Adam hits 20 for a triple. Adam locks out 20 and scores 40 points. (The 1st hit locks out 20, and the 2nd and 3rd hits count for 40 points total.) Bella may no longer score points on 20, but Adam can score on 20 until Bella also locks 20 out.

Input
The first input will be a single integer n indicating how many turns have been taken by each player. Each additional line of input will describe one player's turn, in the following format.

```
playerColor (throw1/throw2/throw3)
```

The format for each throw will be:

```
quantityxvalue
```

Notes:

- The playerColor will always be red or blue
- The quantity will always be 1, 2 or 3
- The value will always be 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, B for Bullseye or M for a miss.

Output
Display the score for the red and blue player in the following format:

```
Red - xxx
Blue - xxx
```

(continued on the next page)

Example Input:
```
3
red (1x5/2x20/3x18)
blue (2x20/3x20/2x18)
red (2x15/2x15/2x15)
blue (2x20/3x15/1x15)
red (3x15/1xM/2xB)
blue (2xB/2xB/1x16)
```

Output to Screen:
```
red - 45
blue - 105
```

F. Column Totals

You are creating a program that reads in numerical values from a comma separated file and finds the total of each column. The data will be integer values, separated by commas.

Input
The first input will be a single integer indicating how many input rows are to follow. Each input row will contain an unknown number of columns, each separated by a comma. When there is no data between two commas or a column does not exist on a row, treat the value of that column as 0.

Output
Display the total of each column, separated by a comma.

Example Input:
```
6
1,3
,,,,,,5
17,5,7,,3,2
-8,18,-2,-3,,9
,,,3,,,,,,,,1
5,5,8,-2,77
```

Output to Screen:
```
15,31,13,-2,80,11,5,0,0,0,0,1
```

This page intentionally left blank

G. Game Collection

You are building a program to sort all your console games. You want them sorted by console, maximum number of players and finally title. The consoles you own are “NES”, “Wii” and “Switch”. All sorting will be done in ascending order.

Input
The first input will be a single integer n indicating how many games are in the collection to follow. The format for each game will be:

```
      gameName/maxNumberOfPlayers/console
```

Output
Display the sorted game list, with each game on its own line, in the following format:

```
      "gameName" (console - maxNumberOfPlayers)
```

Example Input:

```
6
Zelda/1/NES
Zelda/1/Switch
BomberMan/4/Wii
Street Fighter/2/NES
Dig Dug/2/NES
Spatoon/8/Switch
```

Output to Screen:

```
"Zelda" (NES - 1)
"Dig Dug" (NES - 2)
"Street Fighter" (NES - 2)
"Zelda" (Switch - 1)
"Spatoon" (Switch - 8)
"BomberMan" (Wii - 4)
```

This page intentionally left blank

H. Code Names

Secret agents never go by their real name. You are building a program that will take a document containing code names and replace those code names with real names. All names will start with a capital letter.

Note: In the document punctuation may appear before or after agent names.

Input

The first input will be a single integer indicating the number agents to follow The data for each agent will appear on its own line, in the following format:

```
      ^realName,alias1,alias2…,aliasX
```

Following the agent inputs will be a single integer indicating the number of lines in the document to decode.

Output

Display the document with the code names being replaced with the real names of the agents.

Example Input

```
3
^James,Joe,Hax,Billy
^Tina,Charlie,Smoke,Jan
^Blake,Smith,Alex
6
Joe had found out the blacksmith was smuggling goods in January.
Alex located the blacksmith’s supplier and had him arrested. Hax
helped the blacksmith find a new supplier named Smoke. All Smoke’s
goods were setup with a tracker that helped Alex run down Charles,
the king pin that we have been looking for. Charles said, “Alex
you will regret this!”
```

Example Output to Screen

```
James had found out the blacksmith was smuggling goods in January.
Blake located the blacksmith’s supplier and had him arrested. James
helped the blacksmith find a new supplier named Tina. All Tina’s
goods were setup with a tracker that helped Blake run down Charles,
the king pin that we have been looking for. Charles said, “Blake
you will regret this!”
```

This page intentionally left blank

I. Rescue Plan

You are writing a program for a rescue team that frequently has to enter unsafe buildings. To minimize the time they spend on rescue missions they want you to write a program that will give them the fastest route to party that needs be rescued.

When the team gets to a site, they release drones that will map out the building. They want the program to take in the generated map and give them directions to people that need help.

Map Key:
- W – Wall / Rubble
- E – Entrance
- C – Clear Path
- P – Party
- S – Stairs

Your program will need to give directions using the following key:

Directions Key:
- N – North
- E – East
- S – South
- W – West
- U – Up (use stairs to go up one flight)
- D – Down (use stairs to go down one flight)

Note: There will only be one solution that has the fewest number of steps.

Input
The first input will be a two integers `f,r` indicating the number of floors (`f`) and the number of rows (`r`) in each floor in the building. The remaining input lines consist of the building map. After each floor there will be a line of *'s equal to the width of the building.

Output
Display the shortest path to reach the party that needs to be rescued.

(continued on next page)

Example Input

```
4,4
WWCSCC
WWCWWC
SCCCCC
WWCWWS
******
WWPSCW
CCWWCW
SCWCCW
WWWWWW
******
WWWWWW
WWWCCC
SCCCCC
WWWWWS
******
WECCCW
WWCCCC
SCCCWC
WWWWWS
******
```

Example Output to Screen

```
ESSWWUUUEENNEDW
```

J. Mode Finder

You are writing a program that will find the mode(s) of an integer list. A mode is the number that occurs the most in a list.

Input

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will contain an unknown number of integer values, separated by spaces.

Note: every data set has at least 1 value

Output

Display the mode(s) of each input on its own line. When there is more than one mode, separate them by spaces in ascending order.

Example Input

```
4
3 4 5 3 -4 85201 85201 4512 538751
0 1 5 2 5
1
8 4 5 6897456 6897456 8 8 4 5 4 5
```

Example Output to Screen

```
3 85201
5
1
4 5 8
```

This page intentionally left blank

K. Average Fun

You are building a program that will average values in a two-dimensional array. Each element will be a three-digit integer. Follow the below rules to average each cell:

Averaging Rules:

- The first digit will be the average of the first digits of itself and its eight neighbors.
- The second digit will be the average of the second digits of itself and its four diagonal neighbors.
- The third digit will be the average of the third digits of itself and its four orthogonal neighbors (up, down, left, right).
- Always use the original values of the grid to compute averages.
- When computing averages, use integer division.

Input
The input will be a five by five grid of three digit numbers. Each row will be on its own line and each value will be separated by a comma.

Output
Display the averaged grid where each row is on its own line and each value is separated by a comma.

Example Input:
```
222,222,000,222,222
222,222,000,222,222
111,111,111,111,111
123,456,789,123,254
444,234,543,872,999
```

Output to screen:
```
222,111,111,111,222
111,111,111,111,111
121,222,222,131,112
223,234,344,333,344
243,344,434,564,555
```

This page intentionally left blank

L. Complexity

Define the *complexity* of a string to be the number of distinct letters in it. For example, the string `string` has complexity 6 and the string `letter` has complexity 4. You like strings which have complexity either 1 or 2. Your friend has given you a string and you want to turn it into a string that you like. You have a magic eraser which will delete one letter from any string. Compute the minimum number of times you will need to use the eraser to turn the string into a string with complexity at most 2.

Input
The first input will be a single integer indicating how many data sets are to follow. Each data set consists of a single line that contains a single string of at most 100 lowercase ASCII letters ('a'–'z').

Output

Print, on a single line, the minimum number of times you need to use the eraser.

Example Input
```
7
string
letter
aaaaaa
uncopyrightable
ambidextrously
assesses
assassins
```

Output to screen:
```
4
2
0
13
12
1
2
```

This page intentionally left blank

M. NES Controller

You have just learned how print text to the screen and decided to write a program that will display a NES controller.

```
-----------------------------------
|    _              ___________   |
|  _| |_           |           |  |
| |_   _|          | (A)  (B)  |  |
|   |_|    ==  ==  |___________|  |
|                                 |
-----------------------------------
```

Input
None

Output
Displays the ASCII art controller.

Input: none

Output to screen:

```
-----------------------------------
|    _              ___________   |
|  _| |_           |           |  |
| |_   _|          | (A)  (B)  |  |
|   |_|    ==  ==  |___________|  |
|                                 |
-----------------------------------
```

This page intentionally left blank