ACM & WiCS Spring 2021 Programming Contest

March 20th, 2021

- Contest URL: `https://domjudge.cs.fsu.edu`
- You have 5 hours to answer questions.
- You may submit solutions in the following languages:
  - C/C++14
  - Python 3.9.2
  - Java 11
  - C# 7.0
- You are only allowed access to official language documentation and COP3014/COP3363 reference material. You are restricted to:
  - C/C++14: `http://www.cplusplus.com/reference/`
  - Java 11: `https://docs.oracle.com/en/java/javase/11/docs/api/index.html`
  - Python 3.9.2: `https://docs.python.org/3/`
  - C# 7.0 `https://docs.microsoft.com/en-us/dotnet/csharp/`
  - COP3014/COP3363 Reference:
    - `https://www.cs.fsu.edu/~vastola/cop3014/`
    - `https://www.cs.fsu.edu/~vastola/cop3363/`
    - `https://www.cs.fsu.edu/~jayarama/prog1.html`
- You are also allowed one textbook or material no larger than 8.5” x 11” x 2” volume.
- No other resources (e.g. Stack Overflow, Google, Wikipedia) are permitted. Using non-permitted materials will lead to disqualification.
- Teams are restricted to using one workstation (computer) each.
- Use of a cell phone to circumvent these restrictions will lead to disqualification. Use of cell phones in contest rooms is not permitted.
- The Clarifications tab on Domjudge may be used to submit questions pertaining to each problem. Do not use this feature to request troubleshooting help.
- **All input is redirected via STDIN.**
- **All output must be formatted to specification in terms of capitalization and spacing. Please refer to the example output for each question.**
- Do not include a shebang in your submissions.
- **Scoring:**

– Teams are ranked according to score. A higher score is rewarded by answering more questions while acquiring fewer penalties.
– The team that solves the greatest number of questions in the quickest time wins.
– Teams which solve the same number of problems are ranked by least total time.
– Teams may resubmit solutions as many times as needed, but incorrect submission attempts will result in time penalties (and thus a lower score.)
– The scoreboard may be accessed during the first four hours of the contest. The scoreboard will freeze during the final hour.

1 Welcome!

Welcome to the ACM Programming Contest!

This question is to get you used to the judging system we use known as Domjudge. This lets you test how to view questions on Domjudge, how to submit solutions, and familiarize yourself with the question format.

For this question, you will need to read in two integers. Then, you must determine which of the two is larger. You will then print the larger of the two. If the numbers are the same, then you will add them together, and print the sum.

1.1 Input

Please note that all input read into the program is done via STDIN, (e.g. using cin statements in C++).

The input will be a single line with two integers, a and b, separated by a single whitespace.

1.2 Output

Please note that the output to the program should match exactly how it is in the sample output provided. (e.g. Do not prompt for user input, "Please enter the input: ", do not print out things such as "The number is: ", etc.)

If a is larger than b, then print a. If b is larger than a, then print b. If they are the same, they print $a + b$. Your output should be a single line.

1.3 Sample Input/Output

Sample Input 1	Sample output 1
5 10	10
Sample input 2	Sample Output 2
20 15	20
Sample Input 3	Sample Output 3
5 5	10

2 Summing Birds Over the Rainbow

Suzy, a bird watching enthusiast, has made it her mission to find out which location she visits almost every day has the most birds. She carries around her handy-dandy notebook to jot down how many birds she sees at the park, her home, school, and work throughout the week.

There's only one problem...she has to find out which place has the most birds quickly so that she can bird watch with her best friend tomorrow! Can you help her sum up the number of birds at each location?

2.1 Input

The first line will have a single, positive integer less than 100 that represents the number of lines of input to follow.

The following lines will have the first letter of the location (capitalized) followed by a space and then an integer value of the number of birds at that location. You may assume there will always be one location that has the most number of birds.

2.2 Output

The first letter of the location (capitalized) with the most birds.

2.3 Sample Input/Output

Sample Input 1	Sample output 1
3 P 1 H 4 S 2	H
Sample input 2	Sample Output 2
6 P 3 H 2 S 1 W 4 P 4 H 3	P

3 Spring Forward, Fall Back

Spring is arguably the best season, but there's just one thing bad about it: daylight savings time. For whatever reason, society has burdened us with moving our clocks forward an hour in the spring, which means one less hour of sleep that no one mentally benefits from. Fun fact, did you know that daylight savings time was invented because entomologist George Hudson wanted more time to catch bugs? Who could've guessed that someone could be a bigger nerd than a computer science major...

Anyways, to compete with George Hudson's nerdiness, you and your friend decided to make a secret code language based off of the concept *Spring forward, Fall back*, a common phrase used for remembering whether to set your clock an hour forward or backwards on daylight savings day. You and your friend agreed on the following rules for decoding a secret message:

- Start at the first letter of the secret message
- If the current letter is a capital 'S', spring forward a certain number of letters
- If the current letter is a capital 'F', fall back a certain number of letters behind
- If the current letter is not 'S' nor 'F', write it down and go on to the next letter
- The message ends when you reach an exclamation point '!'

After using your secret language with your friend for a while, you decide to write a program that automatically decodes your friend's messages for you.

3.1 Input

The input will be in the following format:

X
Y
string

Where X denotes the number of letters you will spring forward if you encounter an 'S', and Y denotes the number of letters you will fall backwards if you encounter an 'F'. The *string* is the secret message that you are decoding.

You may assume that every *string* contains an '!', and you may also assume that springing forward and falling back will not result in going out of bounds of the *string.*

3.2 Output

Your output will be the decoded message based off of the rules mentioned above:

- Start at the first letter of the string
- If the current letter is not 'S' nor 'F', output it and go on to the next letter
- If the current letter is a capital 'S', spring forward X letters
- If the current letter is a capital 'F', fall back Y letters behind
- The message ends when you reach an exclamation point '!'. Do not output the '!'

3.3 Sample Input/Output

Sample Input 1	Sample output 1
7	secret
6	
seSghet!acrF	
Sample input 2	Sample Output 2
10	RIPspringBreak
12	
RISKingBreaSPsprFunTik!LOL	

4 Surprise Palindromes

Spring is full of surprises! The air gets warmer, the flowers open up, and everyone's happier. You know what isn't a surprise though? Getting a palindrome problem on the ACM contest! However, these aren't your ordinary palindromes.

You will be given a string and you have to determine if the string is a palindrome *when each character is in its ASCII form.* In other words, when every character is converted to its ASCII value, the string of ASCII values reads the same backwards as forwards. Please see the table below for example explanations.

You may utilize any ASCII Table online in order to help you write the program.

4.1 Input

The input is a mixture of characters from the ASCII Table. This includes a combination of letters, numbers, punctuation, whitespace, and various other symbols.

You may assume the input will not be longer than 20 characters, and you may also assume that the input will not contain a null character nor newline character.

4.2 Output

Your output will be 'palindrome' if the string of ASCII values is a palindrome. Otherwise, your output will be 'not palindrome'.

4.3 Sample Input/Output

Sample Input 1	Sample output 1
&BS	palindrome
Sample input 2	Sample Output 2
SUS	not palindrome
Sample input 3	Sample Output 3
5/ {J#	palindrome

Sample I/O 1 Explanation
'&' = 38
'B' = 66
'S' = 83
386683 is a palindrome

Sample I/O 2 Explanation
'S' = 83
'U' = 85
'S' = 83
838583 is not a palindrome

Sample I/O 3 Explanation
'5' = 53
'/' = 47
' '[SPACE] = 32
'{' = 123
'J' = 74
'#' = 35
5347321237435 is a palindrome

5 Snail, Tortoise, and the Hare

It's your classic race between the Tortoise and the Hare...and the Snail. But this time, we are here to find out how these super racers do in different terrains. Given their racing path and their individual speeds on different terrains, compare the racing performances of all the animals in order to find the winner!

The race will require the animals to go on straight paths through 3 possible types of terrain: the Shadow Forest, the Slippery Banks, and the Steep Pastures. Each animal has a different speed when going through the different types of terrain. These speeds are listed in the table below:

	Shadowy Forest (F)	Slippery Banks (B)	Steep Pastures (P)
Snail (S)	6	10	4
Tortoise (T)	7	4	10
Hare (H)	10	6	4

5.1 Input

The first line will have a single, positive integer less than 100 that represents the number of lines of input to follow.

Assume the starting coordinate to be (0,0). The following lines will each have the symbolic letter of the terrain (capitalized) to represent the terrain of the next segment of the race, followed by space-separated X and Y values for the ending coordinate of that segment and starting coordinate of the next segment.

HINT: Remember that the formula for speed is distance/time.

5.2 Output

The first letter of the winning animal (capitalized) of the race. If 2 or more animals have the winning time for completing the race, print out "TIE".

5.3 Sample Input/Output

Sample Input 1	Sample output 1
4 P 40 50 P 40 100 F 70 120 B 200 200	S
Sample input 2	Sample Output 2
2 B 100 100 P 200 200	TIE

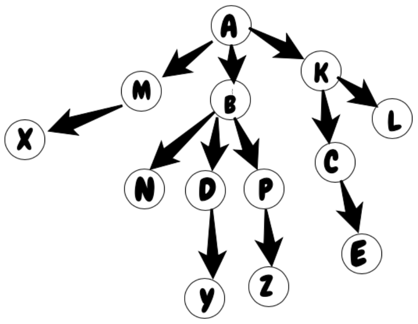
6 Office Madness

Tim is a ruthless go getter who has recently received an MBA from Harvard Business School. He has accepted a job offer from a top Wall Street firm, and wants to receive a promotion as quickly as possible. Being brilliant but lazy, he has decided to sabotage an employee in order to get him fired and thus apply for the open position. He decides his target is Willie, a likable but seemingly incompetent middle manager. Tim wants to undermine Willie by getting his employees fired. Help Tim begin his conquest by listing Willie's most junior employees, given a graph of the firm's employment structure.

For simplicity let us consider the employees are [A-Z]. And the relations among them are provided like the following.

A-M,A-B,A-K,B-D,P-Z,B-P,M-X,K-C,B-N,K-L,D-Y,C-E

The relations are comma separated and A-M means M is working under the supervision of A. Hence we can draw the following graph to represent all the relations provided above.



In the example above, if Willie is worker B, then his most junior employees would be Y and Z.

6.1 Input

The relations along with the target person is provided in the same string. The format of the input string is A-B,B-C,B-D,...,A - the final letter having no relation added to it, is the target person for whom we want to find the subordinates.

Note: The relations are not necessarily provided in descending order.

6.2 Output

The output will be each worker with the lowest positions for the given target. The characters should be printed in upper case, and be alphabetically sorted.

6.3 Sample Input/Output

Sample Input	Sample output
A-M,A-B,A-K,B-D,P-Z,B-P,M-X,K-C,B-N,K-L,D-Y,C-E,B	YZ

7 Florida Man Tries Banking

Florida Man has recently opened a bank in the town of Little Haiduc. Florida Man is a self proclaimed computer science enthusiast, and enjoys boasting of his intimate understanding of data structures, despite being kicked out of the CS program at Shady Dan's Technical School.

The first week of business is a disaster, as Florida Man has implemented a stack structure to manage the customer queue. After consulting with an old classmate, Florida Man implements a priority queue to handle the flow of customers. Unfortunately, customers are randomly assigned a priority, regardless of when they enter the bank. Simulate another wild day at the bank, assuming Florida Man has implemented a minimum priority queue to manage the customers.

7.1 Input

Space separated names (a collection of alphabetic characters) and integers along with two key words: STOP and CLOSE. The input will have the general format of:

integer⟨single space⟩name⟨single space⟩integer⟨single space⟩name⟨single space⟩integer...STOP...a combination of names and integers...CLOSE

The first integer encountered denotes the number of working tellers. Teller numbering starts at 1. Following that integer, the format of the input up until STOP is that of a name and priority, with the integer following a name being said name's priority. These names and integers represent the initial state of the the customer line at the start of business.

Input read after STOP denotes bank activity throughout the day. Any integer read in which is not immediately preceded by a name signifies a teller assisting the next customer at the head of the line. If a name appears in the input, then it signifies a new customer entering the line, and the integer immediately following the name is that customer's priority. The CLOSE keyword signifies that the bank is closed, and no additional input must be read in.

7.2 Output

The output will consist of two portions. Firstly, a listing of the tellers and the name of the customer the teller is assisting. Each teller and name will appear on a separate line in the format. If a teller is not assisting a customer, the word 'available' should be used in place of a customer name. of:

teller⟨single space⟩⟨number⟩:⟨singe space⟩⟨name⟩

The second portion of output will be the individuals remaining in the queue, ordered from the head of the queue to the tail of the queue. The output will be the customer's name, with each individual's name on a separate line.

SAMPLE INPUT AND OUTPUT ON NEXT PAGE

7.3 Sample Input/Output

Sample Input 1

```
2 marco 16 kurtis 12 galen 10 STOP patti 13 2 1 effie 14 ernesto 17 1 CLOSE
```

Sample Output 1

```
teller 1: patti
teller 2: galen
effie
marco
ernesto
```

Sample Input 2

```
3 dorcas 19 lesa 18 rachelle 7 ginger 14 shae 13 elinore 12 STOP joseph 4 zana 5 1 CLOSE
```

Sample Output 2

```
teller 1: joseph
teller 2: available
teller 3: available
zana
rachelle
elinore
shae
ginger
lesa
dorcas
```

8 Spring Cleaning

It's spring cleaning! To help organize your piles of stuff you've created a helper program. This helper organizes each item into a category, provided by you, and alphabetizes each category and the items in them too. Once you've run your super fast helper program you'll be able to go through each category easily and clean up your room in no time!

8.1 Input

Each line of input will be a new item given in the format:

<category name> - <item name>

Where each white space is a single space. If an item is repeated a count should be kept for how many you have.

8.2 Output

Your helper program should present each category in alphabetical order. Each category must be represented in on its own line, with each item reported in alphabetical order and the quantity of that item you have. The format should follow:

<category_1>: <item_1> <item_1_count>, <item_2> <item_2_count>
<category_2>: <item_3> <item_3_count>
etc..

Where all white space values are a single space, and a ',' used to separate multiple items in a category

8.3 Sample Input/Output

Sample Input 1	Sample output 1
clothes - hat clothes - hat misc - painting supplies clothes - shirt	clothes: hat 2, shirt 1 misc: painting supplies 1

Sample Input 2	Sample output 2
clothes - jacket clothes - shirt misc - painting supplies clothes - jacket jewelry - earrings jewelry - necklace jewelry - earrings	clothes: jacket 2, shirt 1 jewelry: earrings 2, necklace 1 misc: painting supplies 1