
CALICO Fall '21 Online

December 18th, 2021

180 Minutes

9 Questions

Problem Packet by CALICO

Table of Contents

Problem Name	Page	Points
Problem 1: Cards Against Programmability	3	5
Problem 2: Back In My Day...	5	5
Problem 3: im stuff	7	7
Problem 4: Commuting in Clumps	9	7
Problem 5: Funny Necklace Numbers	11	11
Problem 6: It's Better Together?	13	13
Problem 7: Edit: Omg thanks for the upvotes!!!	16	13
Problem 8: DOES NOT REACT	19	17
Problem 9: X's and O's, Tic-Tac's and Go's	22	23
Total Possible Points		101

Problem 1: Cards Against Programmability

5 Points

Problem ID: `cards`

Rank: 1

Introduction

Building a house of cards in real life requires cards, patience, and agility—three things that the average programmer doesn't have. We can circumvent these puny physical limitations by making houses of cards digitally!

Your task is to create a program that will display a house of cards with a given number of layers.

Program Input

The first line of the file `cards.in.txt` will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single positive integer n denoting the number of layers the house of cards must have.

Example Input:

```
3
3
1
5
```

Program Output

For each test case, your program should output the corresponding deck of cards based on the following criteria:

- A layer in a house of cards counts as a single row of cards, displayed with alternating slashes / and \ to represent peaks.
- The top row of the house of cards must consist of a single apex /\
- Each layer in the house of cards must have one greater apex than the row above it.
- Each test case output should be separated by a blank line.

Example Output:

 \wedge

Problem Constraints

$$T \leq 100$$

$$0 \leq n \leq 100$$

Problem 2: Back In My Day...

5 Points

Problem ID: age

Rank: 1

Introduction

Roni's grade school math teacher, Mr. Pepper, likes to assign math homework that's more than just arithmetic. This week's homework, for example, is birthday themed—but Roni's not having it. That's why she's asked you for help! However, you don't want to sit through pages and pages of mindless worksheets, so you've decided to get some help from your trusty computer.

Your task is to create a program that will take in two people's ages and output how long it will be before one is twice as old as the other.

Program Input

The first line of the input from STDIN will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line in the format: `<NAME1> <AGE1> <NAME2> <AGE2>`
 - `<AGE1>` and `<AGE2>` will be represented as positive integers.

Example Input:

```
3
Amanda 4 Brian 1
Christine 6 Daniel 5
Eric 3 Fregory 21
```

Program Output

For each test case, your program should compute how long it will take for one person to be twice as old as the other. Your output should be created based on the following criteria:

- Your output should be in the following format:

```
In <n> year(s):  
<NAME1> will be <FINAL1> and <NAME2> will be <FINAL2>
```

- If nobody will ever be twice as old as the other, your program should output `Time will end before you are happy`
- Each test case output should be separated by a blank line.

Example Output:

```
In 2 year(s):  
Amanda will be 6 and Brian will be 3
```

```
Time will end before you are happy
```

```
In 15 year(s):  
Eric will be 18 and Fregory will be 36
```

Problem Constraints

$$T \leq 1000$$

$$1 \leq \text{<AGE1>}, \text{<AGE2>} \leq 1.0 \times 10^8$$

Assume nobody is currently twice as old as the other.

Problem 3: im stuff

7 Points

Problem ID: hi

Rank: 1

Introduction

Dad jokes are the gold standard for comedy. Nothing causes the nose the exhale more than the timeless comment "Hi Hungry, I'm Dad." That being said, you're sick of constantly hearing your dad's eye roll-inducing remarks at the dinner table—so you decide to create a program to beat him at his own game.

Your task is to create a program that will a dad joke from the given statement.

Program Input

The first line of the input from STDIN will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line containing a statement in the following format:
`<...> <KEYWORD> <MESSAGE>`
- `<KEYWORD>` consists of the word `I 'm` with any potential capitalization or apostrophe use.

Example Input:

```
3
dad Im hungry
THATS IT IM LEAVING YOU AND TAKING THE KIDS
honey i'm pregnant
```

Program Output

For each test case, your program should correctly output the corresponding dad joke based on the following criteria:

- Your output should be in the following format: `Hi <MESSAGE>, I'm dad.`
 - `<MESSAGE>` should consist of the `<MESSAGE>` following the `<KEYWORD>` in the input statement, while preserving capitalization, punctuation use, and spacing.

Example Output:

`Hi hungry, I'm dad.`

`Hi LEAVING YOU AND TAKING THE KIDS, I'm dad.`

`Hi pregnant, I'm dad.`

Problem Constraints

$T \leq 100$

Assume the input does not contain more than one valid `<KEYWORD>`

Assume that both a `<KEYWORD>` and `<MESSAGE>` exist in every input.

The input statement may contain multiple special characters or numbers.

Assume the input statement has normal spacing.

Problem 4: Commuting in Clumps

7 Points

Problem ID: `commute`

Rank: 1

Introduction

As the top traffic inspector in your region, you're looking at highways and avenues all day long. However, one road you'd rather not work with is the single-lane highway connecting all the neighborhoods in your town. One day, on the way to work, you notice all the different types of drivers in front and behind you: the slow ones, the fast ones, and the idiots who think the shoulder lane is for passing. However, you notice one additional thing: the car in front of you is going way slower than you'd like, and it's slowing down all the cars behind you! That causes you to think: how do slow drivers affect traffic?

Your task is to create a program that will output the number of traffic clumps that will form given cars' maximum driving speeds.

Program Input

The first line of the input from STDIN will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line consisting of two values separated by a space. The values are as follows:
 - A positive integer n denoting the number of cars on the road.
 - A comma-separated sequence of n non-negative integers $s_1..s_n$, denoting each car's maximum driving speed. The car in front's maximum speed is listed on the left, and the last car's maximum speed is listed on the right.

Example Input:

```
3
3 4,5,3
5 1,1,1,1,1
10 19,34,24,20,26,15,38,22,45,12
```

Program Output

For each test case, your program should correctly output the number of traffic clumps formed according to the following criteria:

- A clump consists of a group of consecutive cars all going the same speed.
- Cars will drive their maximum driving speed unless the driver in front of them is driving slower. In that case, the car would match their speed to avoid a collision.

Example Output:

2
1
3

Problem Constraints

$$T \leq 1.0 \times 10^6$$

$$1 \leq n \leq 1.0 \times 10^8$$

$$0 \leq s_{1..n} \leq 1.0 \times 10^8$$

Problem 5: Funny Necklace Numbers

11 Points

Problem ID: `necklace`

Rank: 2

Introduction

You are playing a game with beads on bracelets. Each bead has a number, and the bracelet can only be rotated around your wrist (not flipped or reversed or otherwise rearranged). You gain points based on how you orient your bracelet around your wrist, starting at a set spot on your wrist and reading the beads left to right to form a single integer. You want to know the maximum number of points you can gain by rotating the bracelet around the set point.

Your task is to create a program that will take in a number and output the highest possible number attainable by rotating it in place.

Program Input

The first line of the input from STDIN will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A single line containing a positive integer n

Example Input:

```
4
12
6493
9919
100
```

Program Output

For each test case, your program should output the largest possible number attainable by rotating the current number in place. For example, rotating "123" one to the right will yield the number "312." You should not include leading zeros.

Example Output:

```
21
9364
9991
100
```

Problem Constraints

$T \leq 100$

$0 < n \leq 1.0 \times 10^8$

Problem 6: It's Better Together?

13 Points

Problem ID: `allergy`

Rank: 2

Introduction

After suffering an unfortunate allergic reaction from a stray cucumber at your local organic farmer's market, you've been worried about what other things you may be allergic to. You didn't want to pay for a skin-prick allergy test, so you decide to go the DIY route—by exposing yourself to potential allergens one-at-a-time while recording any side effects. However, the possibility of suffering through numerous reactions for weeks has you feeling nauseous. Your friend Franjessica suggests another way: grouped-testing—grouping multiple tests into batches, and recording one result for each batch. You can save both time and money; what's not to love?

Your task is to create a program that will output potential allergens given a set of grouped-testing results.

Program Input

The first line of the input from STDIN will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A first line consisting of two numbers separated by a space. The numbers are as follows:
 - A positive integer n denoting the number of grouped-tests that will follow.
 - A positive integer m denoting the number of items that can be tested.
- The next n lines will each consist of a grouped-test, represented by three values separated by spaces. The values are as follows:
 - A positive integer $i_{1..n}$ denoting the number of items tested at once.
 - A string representing the result of the test, shown as either `POSITIVE` or `NEGATIVE`
 - A forward-slash-separated list of names $i_{1..n}$ long, representing the items tested.
- A blank line separating individual test cases.

Example Input:

```
4
3 4
dish_soap sponges water ceramic_plates
4 POSITIVE dish_soap/sponges/water/ceramic_plates
2 POSITIVE dish_soap/sponges
1 POSITIVE dish_soap

2 4
cardboard_box pizza grated_parmesan crushed_red_pepper
2 NEGATIVE cardboard_box/pizza
3 POSITIVE pizza/cardboard_box/grated_parmesan

1 2
air dust
2 POSITIVE air/dust

4 7
tomato kumquat watermelon jalapeno jackfruit cucumber bell_pepper
3 POSITIVE cucumber/bell_pepper/tomato
3 NEGATIVE jalapeno/tomato/kumquat
3 NEGATIVE bell_pepper/watermelon/jalapeno
3 POSITIVE watermelon/jackfruit/bell_pepper
```

Program Output

For each test case, your program should correctly output the potential allergens according to the following criteria:

- Your output should be in the format:

```
KNOWN allergens: <ITEM>/<ITEM>/<...>
POTENTIAL allergens: <ITEM>/<ITEM>/<...>
```

- A test shows positive if any one of the items tested is an allergen. A test shows negative if none of the tested items are allergens.
 - Items should only be classified as known allergens if it is impossible for them to be otherwise. An item should be classified as a potential allergen if their status is not guaranteed.

- Names in each list should be sorted in alphabetical order. Underscores should be considered as last in alphabetical order.
- If a classification has no applicable items, your program should insert `NONE` in place of the item list.
- Each test case output should be separated by a blank line.

Example Output:

```
KNOWN allergens: dish_soap
POTENTIAL allergens: ceramic_plates/sponges/water
```

```
KNOWN allergens: grated_parmesan
POTENTIAL allergens: crushed_red_pepper
```

```
KNOWN allergens: NONE
POTENTIAL allergens: air/dust
```

```
KNOWN allergens: cucumber/jackfruit
POTENTIAL allergens: NONE
```

Problem Constraints

$$T \leq 1.0 \times 10^6$$

$$1 \leq n \leq 1.0 \times 10^5$$

$$1 \leq i_{1..n} \leq m \leq 1.0 \times 10^5$$

Item names will only consist of lowercase letters and underscores.

Problem 7: Edit: Omg thanks for the upvotes!!!

13 Point(s)

Problem ID: `social`

Rank: 2

Introduction

All your ideas have failed. Over and over. Turns out, nobody wants a dating app only for catfishers, a pregnancy test app, or "another programming contest." So screw it! Let's just copy Tumblr and Reddit. Where better to start than with their iconic comment formatting system?

Your task is to create a program that will output a formatted comment thread given a list of comments and their replies.

Program Input

The first line of the input from STDIN will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A first line containing a single positive integer n denoting the number of comments on the comment thread.
- The next n lines each represent a posted comment in the format:
`<COMMENT-ID> <REPLY-ID> "<MESSAGE>"`
 - `<COMMENT-ID>` represents the comment's unique ID.
 - `<REPLY-ID>` represents the ID of the comment the post is replying to. If the post is not replying to a comment, `<REPLY-ID>` will be shown as `n/a`
 - `<MESSAGE>` represents the contents of the post.
 - Comments are listed in the order in which they are posted. The comment posted first is listed at the top, and the comment posted last is listed at the bottom.
 - A comment can only reply to a comment listed above it, or none at all.
- A blank line separating individual test cases.

Example Input:

```
3
4
C-A n/a "apple juice > orange"
C-B C-A "orange juice taste like battery acid but you do you"
C-C C-B "cranberry >"
C-D n/a "some dude really said cranberry >"

4
a123 n/a "[deleted]"
b234 a123 "[removed]"
c345 a123 "This comment was removed automatically. Learn more here"
d456 b234 "[deleted]"

6
p1105-20 n/a "Orecchiette is the best pasta shape. This looks awesome"
a1106-20:m p1105-20 "Thank you so much"
a1106-20 n/a "Do you make a roux for the sauce? This looks amazing."
a1107-20 p1105-20 "nah man, fusilli all the way"
p1107-20:m a1107-20 "Fusili is just the worst"
p1109-20 a1106-20 "No roux here. Just milk, cream and parm"
```

Program Output

For each test case, your program should correctly output the formatted comment thread according to the following criteria:

- Each comment should only have its corresponding `<MESSAGE>` displayed.
- A post replying to another comment should be displayed underneath the parent comment, preceded with an additional indent. A reply to a reply should have two additional indents, and so on.
 - Indents are represented with a pipe character `|` followed by a space.
 - If replies already exists under the parent comment, the post should be listed below the most recent reply.
- A comment not replying to anything should not have any indents.
- Parent comments of the same indent-level should be sorted in chronological order.
- Each test case output should be separated by a blank line.

Example Output:

```
apple juice > orange
| orange juice taste like battery acid but you do you
| | cranberry >
some dude really said cranberry >

[deleted]
| [removed]
| | [deleted]
| This comment was removed automatically. Learn more here

Orecchiette is the best pasta shape. This looks awesome
| Thank you so much
| nah man, fusilli all the way
| | Fusili is just the worst
Do you make a roux for the sauce? This looks amazing.
| No roux here. Just milk, cream and parm
```

Problem Constraints

$T \leq 1.0 \times 10^6$

$1 \leq n \leq 1000$

Assume every <COMMENT-ID> is unique.

Assume no <COMMENT-ID> is n/a

Assume no <COMMENT-ID> or <REPLY-ID> contains spaces.

Assume no <MESSAGE> contains quotes.

Problem 8: DOES NOT REACT

17 Points

Problem ID: `react`

Rank: 3

Introduction

You thought OChem was hard? In the imaginary field of Binary Chemistry (otherwise known as BChem), scientists study the relationships between two families of elements: Right Elements and Left Elements. Elements from the same family will react with each other, but elements from opposite families will not (so Rights will not interact with Lefts). You're starting your career in BChem at a local laboratory; you're in charge of discovering new reactions between elements. You're given a handy reference sheet with currently known reactions to help you out, but that's when you realize: none of the elements' families are labeled—just their known reactions. Hopefully it's not too late to switch laboratories...

Your task is to create a program that will determine if two elements will react, given a list of known reactions.

Program Input

The first line of the input from STDIN will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A first line containing the two target elements in the format: `<X>` and `<Y>`
- A second line containing a single non-negative integer n denoting the number of known reactions that follow.
- The next n lines each consist of a single known reaction in the following format:
`<A>` `<REACTS WITH/DOES NOT REACT WITH>` ``

Example Input:

```
4
Fecrothil and Agloetium
4
Dospusten REACTS WITH Ostuysium
Ostuysium DOES NOT REACT WITH Fecrothil
Dospusten DOES NOT REACT WITH Fecrothil
Ostuysium DOES NOT REACT WITH Agloetium

Ostuysium and Dospusten
0

Ost and Fec
8
Dos REACTS WITH Dron
Spo DOES NOT REACT WITH Dos
Agloe REACTS WITH Spo
Dos REACTS WITH Fec
Agloe DOES NOT REACT WITH Skio
Dron DOES NOT REACT WITH Agloe
Dron REACTS WITH Skio
Skio REACTS WITH Ost

Agloetium and Spoenyx
5
Fecrothil REACTS WITH Agloetium
Dospusten REACTS WITH Ostuysium
Ostuysium DOES NOT REACT WITH Fecrothil
Dospusten DOES NOT REACT WITH Agloetium
Ostuysium REACTS WITH Spoenyx
```

Program Output

For each test case, your program should output the reactivity of the two elements based on the following criteria:

- Your output should be in the following format:

<X> <REACTS WITH/DOES NOT REACT WITH> <Y>

- If not enough information is given to determine the two elements' reactivity, your program should output UNKNOWN

Example Output:

```
Fecrothil REACTS WITH Agloetium  
UNKNOWN  
Ost REACTS WITH Fec  
Agloetium DOES NOT REACT WITH Spoenyx
```

Problem Constraints

$T \leq 100$

$0 \leq n \leq 1.175 \times 10^4$

Assume every reaction involves a unique pair of elements.

Problem 9: X's and O's, Tic-Tac's and Go's

23 Points

Problem ID: `tictacgo`

Rank: 3

Introduction

You and Peater often get bored in your APUSH class, so you both make up games to pass the time. Tic-Tac-Toe gets boring quick, and so does Dots and Boxes, and so does Connect 4, and so does... a lot of other games. Yet, one game has remained interesting throughout hours and hours of history lectures: Tic-Tac-Go (a marriage between the games Tic-Tac-Toe and Go), where you try to get a certain number of pieces in a row—but you can also surround, capture, and convert your opponent's pieces. However, the process of converting pieces requires a lot of erasing and rewriting, so you want to write a program to automate the process!

Your task is to create a program that will output the final state of a Tic-Tac-Go grid after a given piece is placed. You do not need to account for win/lose states.

Program Input

The first line of the input from STDIN will contain a positive integer T denoting the number of test cases that follow. Each test case will have the following input:

- A first line consisting of grid dimensions and a target coordinate, separated by a colon. The two values are as follows:
 - The dimensions consist of two positive integers r and c separated by an \times , denoting the number of rows and columns, respectively.
 - The target coordinate consists of two numbers within parentheses separated by a comma. The numbers are as follows:
 - Two positive numbers r_T and c_T representing the row and column of the next X placed within the grid.
 - The coordinate $(1,1)$ represents the top left corner, with additional rows going down and additional columns going right. For example, the coordinate $(3,1)$ represents the position at the third row and at the very left.

- A grid with the given dimensions containing any of the following three characters:
 - An \times denoting a grid space occupied by your piece.
 - An \circ denoting a grid space occupied by your opponent's piece.
 - A dash – representing empty space.

Example Input:

```
5
2x2: (1, 2)
--
XO
```

```
4x5: (3, 3)
X-XOO
OXOOO
-X-OO
OXOOO
```

```
7x2: (6, 1)
-O
-X
O-
O-
OO
--
O-
```

```
5x6: (3, 4)
-----
--XO--
-XO-O-
--XO--
-----
```

```
5x4: (3, 3)
--O--
-OXO-
OX-XO
-OXO-
--O--
```

Program Output

For each test case, your program should output the final state of the grid after the next \times piece is placed. Your output should be created based on the following criteria:

- Same-side pieces directly adjacent to each other are defined to be part of the same clump.
 - Adjacent spaces are defined as the spaces directly above, below, to the right, and to the left of a piece.
- A clump of pieces is defined as surrounded when all pieces within the clump have their adjacent spaces occupied by opposition pieces, grid borders, or same-side pieces. In other words, a clump is surrounded when it is impossible for it to expand in size.
 - If a clump of pieces is surrounded (at least in part) by opposition pieces, all the pieces within the clump are converted to opposition pieces (\times to \circ , and vice versa).
 - All clumps made of the same piece should be converted simultaneously (at the same time).
 - The player who placed the last piece has priority in converting opponents' clumps, meaning any surrounded \circ clumps should be converted to \times pieces first, before the opposite happens.
- A turn ends (described as the final state of the grid) when no clumps on the grid are surrounded (at least in part) by opposition pieces. This may include the player's own pieces.

Example Output:

-X

XX

X-XXX

OXXXX

-XXXX

OXXXX

-O

-X

O-

O-

OO

X-

O-

--XO--

-XXXO-

--XO--

--O--

-OOO-

OOOOO

-OOO-

--O--

Problem Constraints

$T \leq 100$

$1 \leq r, c \leq 50$

Assume the input grid is already in a final state.

Assume the target coordinate corresponds to an empty space.