2021 ACM Programming Contest

- Complete as many questions as you can during the contest.
- All questions require you to read the test data from standard input and write results to standard output; you cannot use any non-contest files for input or output.
- The input to problems will consist of multiple test cases unless otherwise noted.
- Programming style is not considered in this contest. You are free to code in whatever style you prefer. Commenting code is not required.
- All communication with the judges will be handled by the PC2 environment.
- Allowed programming languages: C, C++, Java, and Python.
- **Netbeans users: Remove package statement before you submit**
- **Java users: remember to call `nextLine()` on a scanner after reading in an integer when needed.**
- All programs will be re-compiled prior to testing with the judges' data.
- Output must match the sample format exactly. Do not print prompts unless asked to.
- Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and string libraries are allowed. The standard Java API is available and can be used, except for those packages that are deemed dangerous by contest officials (e.g., that might generate a security violation)
- Hardcopy reference materials are allowed, but not electronic material. Calculators are fine, but no laptops, phones, USB drives, etc. are allowed.
- No connection to the Internet may be made during the contest.
- **Only one computer may be used per team.**

Judges decisions are to be considered final. No cheating will be tolerated.

Problem A - Eek!

Your friend has an intense fear of certain words, but only one word at a time (you have kind of weird friends). If a sentence contains the word your friend is afraid of, regardless of spacing or capitalization, your friend will yell 'eek'. Otherwise, your friend will just read the sentence and say 'meh'.

Write a program that reads in the word your friend is currently afraid of and a set of sentences, and output whether they will say eek or meh after reading each sentence.

Input The first line of input will be an integer, $1 \leq n \leq 100$, the number of lines in the input. The next line will contain a single word, the word your friend is afraid of. The next $n - 1$ lines will contain one sentence each.

Output Output should contain $n - 1$ lines, with either *eek* or *meh* on each line.

Sample Input

```
5
pumpkin
I don't understand the appeal of pumpkin spiced lattes.
They are not really that good.
They also don't really taste like pumpkin.
Pumpkin pie is the only good way to consume pumpkin.
```

Sample Output

```
eek
meh
eek
eek
```

Problem B - Candy Counting

The neighbor kid from across the street is taking stock of her trick-or-treating haul. She has drafted you as her accountant. She pulls each piece of candy out of her bucket one at a time and tells you what it is. Your job is to let her know which type of candy she has the most of.

Input The input will begin with a positive integer representing the number of pieces of candy, $n \leq 100$. The next n lines will contain the name of each type of candy, one per line.

Output The output should be the name of the type of candy that appears most frequently in the list. There will always be a single type of candy that appears most frequently (i.e. there will never be any "ties").

Sample Input

```
10
Kit-kat
Snickers
Sweet Tarts
Reeses Peanut Butter Cup
Snickers
Kit-kat
Snickers
Kit-kat
Snickers
Snickers
```

Sample Output

```
Snickers
```

Problem C - Forbidden Forest

While running away from a mad clown with a balloon-saw (a saw that only damages balloons), you approach a dark forest. At the trailhead you see a sign with a QR code for trail information which you of course stop and scan. The website for these trails includes quite a bit of information like the slope and distance (along the trail) of each section of the trail. However, in your hysterical (and soon to be balloon-sawed) state, you need to know the total elevation change of each trail to determine whether you have the energy to make it through the forest (alive).

For example, data about a specific trail will state that it is completely flat for 500 meters, then switch to an 8° incline for 1000 meters, and then switch to a 2° decline for the final 500 meters.

The really scary part... Turns out the clown was NOT carrying a balloon-saw but instead a balloon saw (a chainsaw made out of balloons)...

Input Input begins with a positive integer $n \leq 100$ indicating the number of trail sections. The following $2 * n$ lines describe the trail sections from the beginning to the end of the trail. Each trail section is described by two integers: a and d, ($-50 \leq a \leq 50$, $1 \leq d \leq 10000$) indicating its angle of elevation in degrees and the distance in meters along that section of the trail, respectively.

Output Display the difference in elevation between the start of the trail and the end of the trail in meters. The total difference in elevation will always be greater than 0 meters. Round answers to the hundredths place. Always print answers to two decimal places and include the leading 0 on answers between 0 and 1.

Sample Input

```
3
0
500
8
1000
-2
500
```

Sample Output

```
121.72
```

Problem D - Don't Trip!

Look out! You are being chased by a chainsaw-wielding axe murderer. Both you and the axe murder run at a constant rate. The ground is muddy and you trip once every minute. When this happens you get back up and continue running, but the mud slows you down by 10%. However, you do have a head start on the axe murderer. The axe murderer is unaffected by the mud. Your task is to write a program that figures out how many minutes you have to live.

Input The input will consist of three integers (one per line, in the following order):

- X : The number of feet per minute the axe murderer moves
- Y : The number of feet per minute you move
- Z : The number of feet you are away from the axe murderer when he notices you

For example:

```
10
12
5
```

In this case, there is one test case, and $X = 10$, $Y = 12$, and $Z = 5$.

Output You should output the number of minutes you have to live, always rounded DOWN.

For the previous example, the output should be 5. To see this, consider the table below, which shows the position in feed of you and the axe murderer.

Minute	Murderer	You
0	0	5
1	10	17
2	20	27
3	30	36
4	40	44
5	50	51
6	60	58

So you lived 5 full minutes before the murderer reached you.

Problem D - Don't Trip! (cont.)

Sample Input

```
10
12
5
```

Sample Output

```
5
```

Problem E - Chain Reaction

Mikey is very picky about his candy and has established a set of rules for when to trade with other kids:

1. 5 bags of candy corn (C) is worth 2 Kit-Kats (K)
2. 4 Kit-Kats (K) are worth 2 Reese's cups (R)
3. 3 Reese's cups (R) are worth two Snickers bars (S)

Mikey never has trouble finding kids in his neighborhood willing to trade with him, and he keeps trading as long as any of his three rules can be applied. Write a program that reads in an integer specifying the number of test cases.

Input The first line of the input, n, denotes the total number of lines in the input.

Each test case is contained on a single line, comprised of a set of letters: C, K, R, and S. Each denotes a corresponding piece of candy.

If more than one rule applies at the same time, they are always executed in the order specified above.

Here is an example:

`CCKKCCSCRRR`

First the candy corn would be traded for two Kit-Kats: `KKKKSRRR`

Then the four Kit-Kats would be traded for two Reese's cups: `RRSRRR`

Next, three Reese's cups would be traded for two Snickers bars: `SSSRR`

At that point no more rules apply, and Mikey ended up with five pieces of candy, so the program would output 5.

Output For each test case, output a line with an integer representing the number of pieces of candy Mikey has remaining after completing all of this trades.

Sample Input

```
2
CCKKCCSCRRR
KKSRKRCCSKRS
```

Sample Output

```
5
9
```

Problem F - Lazy Trick-or-Treating

The neighborhood kids are lazy and came up with a plan to get a lot of candy during trick-or-treating without having to walk as far: they visit the same houses over and over. The people giving out the candy have started to catch on though – if not enough other kids come to a house in between the lazy kids' visits, they are noticed.

Your task is to write a program that reads in a number x and a list of the costumes of each trick-or-treater visiting a house. If the same costume appears twice without at least x other costumes in between, that kid is busted (and your program should print out the costume followed by the words `is busted!`). For this problem you're going to assume that every trick-or-treater has a different costume. Also note that the same kid can get busted more than once.

Input The starts with a positive integer x indicating the number of other trick-or-treaters that must visit a house in order for repeat not to be noticed. This will be followed by another postive integer v indicating the total number of trick-or-treaters that visit the house.

Then there will be a series of v strings containing the costumes of the trick-or-treaters, one per line.

Output The program should print out `<costume> is busted!` whenever a costume appears again without at least x other costumes in between.

Sample Input:

```
3
10
zombie
ghost
goblin
zombie
goblin
fairy
fireman
ghost
ghostbuster
fairy
```

Sample Output:

```
zombie is busted!
goblin is busted!
```

Problem G - Candy Bucket

You're going trick-or-treating! At each house you receive a piece of candy. As you go house to house, sometimes you eat a piece of candy. At the end of the night, you tip out your candy bucket to see what's left.

Every time you receive a piece of candy, you put it on top of your other pieces. Whenever you eat a piece of candy, you grab the first one you can reach. At the end of the night, you tip your bucket over so that what's at the bottom of the bucket is at the top. Your job is to list, in order, the contents of the bucket afer dumping it out.

Input The input will specify if you receive a piece of candy or if you eat the first candy on top of the pile. You will never try to eat candy when you don't have any.

The first line of the input will be an integer, $3 \leq n \leq 13$, the number of stops. The stops are represented on the following n lines with a single string on each line. The string will either be the name of the candy or *eat*.

Output Your program should compute the resulting list of candy in the correct order, as described above, and print this information out.

Problem G - Candy Bucket (cont.)

Sample Input

```
10
Reese's Cup
Snickers
eat
M&M's
Sour Patch Kids
Tootsie Pop
Skittles
Snickers
eat
eat
```

Sample Output

```
Reese's Cup
M&M's
Sour Patch Kids
Tootsie Pop
```

Problem H - Monster Hunting

You're a monster hunter! Monsters are overrunning Wright State, and it's your job to stop them. There are 5 kinds of monsters: zombies, mummies, ghosts, vampires, and hobgoblins. You and your fellow students decide to make a game out of how many monsters you can eliminate in an hour. You decide that zombies are worth 1 point, mummies are worth 2 points, ghosts are worth 4 points, and vampires are worth 8. Hobgoblins, being friendlly goblins, are worth -10 points.

After an hour, you tally up your scores to see who won. Your task is to create a program that will determine who won, given a list of what monsters you and the other students got.

Input The input will first specify the total number of input lines.

The next line will contain an integer representing the number of contestants, c. The next c lines contain names of contestants, where each line contains only one name.

The next c lines will contain strings that define how many of each monsters each contestant killed. Monster lines will be comprised of an integer followed by a string denoting the type of monster. Z for zombies, M for mummies, G for ghosts, V for vampires, and H for hobgoblins. There is no particular order monsters will appear in the line, and monsters might show up twice in one line. If a contestant does not kill a particular type of monster, it will not appear in the list. An example line might be:

```
1Z3G4M2V1H1Z
```

Which would score 28 points.

The first line of monsters will correspond with the first contestant named, the second line of monsters will correspond with the second contestant named, and so on.

Output Output the name of the contestant who won by scoring the most points. There will be no cases that produce a tie or a negative score.

Problem H - Monster Hunting (cont.)

Sample Input

```
6
2
Dr. Cheatham
Dr. Doom
1Z4G
5M1H
```

Sample Output

```
Dr. Cheatham
```

Problem I - Football Scores

In monsterball, teams generally score ghostups (worth 7 points), or flap goals (worth 3 points). Any number or combinations of ghostups and flap goals may be scored in one game. Your task is to determine how many ways (permutations, not combinations) a team could have scored a particular score.

Input The first line of the input will be the total number of input lines. The remaining lines will each contain one integer, the score of a game.

Output Given each input score, output the number of combinations of ghostups and flap goals, n, that will achieve the input score.

Sample Input

```
6
7
3
5
10
15
23
```

Sample Output

```
1
1
0
2
1
10
```

Problem J - Jigsaw Puzzle

Hello programmer, I want to play a game. The rules are simple, I will show you a jigsaw puzzle and you will tell me how many pieces you have. The puzzle will be transferred to you in the form of a two dimensional grid of characters, either a `.` character to represent part of a piece or a `#` character representing a boundary between the pieces.

A piece consists of a set of `.` characters that are adjacent horizontally or vertically but not diagonally. For example the puzzle below has five pieces, one in each corner and then another one in the middle.

```
...#...
...#...
###.###
...#...
...#...
```

Hurry up programmer, you're running out of time!

Input The first line of input will consist of two numbers m and n separated by a single space specifying the size of the puzzle. The next m lines will consist of n characters each, each character being either `.` or `#`.

Both m and n will both be greater than or equal to 1 and less than 1024.

Output Your output must consist of a single integer specifying the number of pieces in the puzzle. The number of pieces will be at least one and no more than 65536.

Problem J - Jigsaw Puzzle cont.

Sample Input

```
3 3
..#
.#.
##.
```

Sample Output Output 1:

```
2
```

Problem K - The Ring

You are the head wall engineer in the zombie apocalypse and are currently reviewing potential designs for new rings of walls to build around your settlement. The proposed designs are submitted in the form of a two dimensional grid of characters with a `.` representing empty space and a `#` representing a segment of wall. Designs are judged based on two criteria: whether they form a closed loop (preventing zombies from entering) and if they waste materials (every minute outside the walls gathering materials is dangerous). A design has wasted materials if there are any segments of the wall that aren't part of the main loop or if any segment of the wall in the main loop has more than two adjacent wall segments. For example,

```
..##.
.#.#.
..#..
```

is an invalid design because the left segment in the top most row has three adjacent segments and wastes materials.

Input The first line of input will be two integers m and n, $1 \leq m, n \leq 1024$, separated by a single space specifying the dimensions of the proposes plan. The next m lines will have n characters each and all characters will be either a `.` or a `#`.

All plans will contain at least one segment.

Output Your program should print out `valid` if the proposed design is valid and `invalid` if it is not.

Problem K - The Ring (cont.)

Sample Input

```
5 5
..#..
.#.#.
#...#
.#.#.
..#..
```

Sample Output

```
valid
```

Problem L - Witches Coven

Its common sense to a witch: Join a coven as soon as you can.

Why? What a mortal question.

Witches gain power for each witch they bring into the coven! Additionally, they gain power from other witches like a pyramid scheme.

Each witch has a spooky power level of 1 on their own. For every witch they personally bring in, they gain 0.5 (1/2) of their power. Should that brought-in-witch bring in a friend, the brought-in witch gains power, and the original witch gains a portion of the new power gained, albeit at a dimished rate the further the added witch is away from the original witch.

To find the power of any one witch you must find the power level of all the witches brought in under this pyramid scheme below the witch in question. The rate of drop off in power is 1 divided by the number of jumps from the witch in question to a lower-tier witch + 1, so a witch brought in directly will give 1/2 of its power while a witch one further hop away will only give 1/3 of its power.

Given some witch coven, find the strength of all the witches.

Constraints:

- The coven is a directed tree
- There will never be more than 13 witches in a coven
- All witches in the coven will have at least a cardinality of 1

Input Expected input (in order of lines):

- Num test cases / covens you will need to calculate for
- Number of witches in a given coven
- Flattened adjacency matrix
- Number of witches in a given coven
- Flattened adjacency matrix

Input Constraints:

- Where N is the number of witches, The input adjecency matrix is NxN in length
- There will be no more than 13 witches in a Coven
- The adjacency matrix is unweighted.
- The adjacency matrix will always be a single connected component.

Output A list of all the power levels of the witches per coven.

Output Constraints:

- The order of the witch power levels should be in the same order as the witches are in the adjacency matrix.
- All witches have a base power level of 1.
- The effective power level increase any one witch contributes is a function of the power of the sub-coven(s) beneath that witch.

Sample Input

```
1
6
0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0
```

In this example, there is (1) that has (6) witches in it.

Sample Output

```
[2.7083333333333335, 1, 1, 1.6666666666666665, 1.5, 1]
```