2019 University of Virginia High School Programming Contest

Welcome to the 2019 University of Virginia High School Programming Contest. Before you start the contest, please be aware of the following notes:

Rules

1. There are twelve (12) problems in this packet, using letters A-L. These problems are *loosely* sorted by difficulty. As a team's solution is judged correct, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	Legion Roster	red
B	Ceasar Cipher	light purple
C	Target Practice	yellow
D	Gods Abound	light blue
E	Fortune Teller or Fraud?	pink
F	Circus Sorting	light green
G	Pesky Publicans	orange
H	Et tu brute?	gold
I	Marching Legion	dark blue
J	Convivium Combat	white
K	Casino Romale	dark purple
L	Aqueduct Rider	dark green

2. Solutions for problems submitted for judging are called runs. Each run will be judged.

   The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
Yes	Your submission has been judged correct.
No - Wrong Answer	Your submission generated output that is not correct or is incomplete.
No - Output Format Error	Your submission's output is not in the correct format or is misspelled.
No - Excessive Output	Your submission generated output in addition to or instead of what is required.
No - Compilation Error	Your submission failed to compile.
No - Run-Time Error	Your submission experienced a run-time error.
No - Time Limit Exceeded	Your submission did not terminate within one minute.

3. A team's score is based on the number of problems they solve and penalty minutes, which reflect the amount of time and number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty minutes are issued equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty minutes are added for problems

that are never solved. Teams are ranked first by the number of problems solved and then by the fewest penalty minutes.

4. This problem set contains sample input and output for each problem. However, the judges will test your submission against several other more complex datasets, which will not be revealed until after the contest. One challenge is designing other input sets for yourself so that you may fully test your program before submitting your run. Should you receive a "wrong answer" judgment, you should consider what other datasets you could design to further evaluate your program.

5. In the event that you think a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem carefully before requesting a clarification. If the judges believe that the problem statement is sufficiently clear, you will receive the response, "The problem statement is sufficient; no clarification is necessary." If you receive this response, you should read the problem description more carefully. If you still think there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found. If the problem statement is ambiguous in specifying the correct output for a particular input, please include that input data in the clarification request.

   You may not submit clarification requests asking for the correct output for inputs that you provide. Sample inputs may be useful in explaining the nature of a perceived ambiguity, e.g., "There is no statement about the desired order of outputs. Given the input: . . . , would not both this: . . . and this: . . . be valid outputs?".

   If a clarification that is issued during the contest applies to all the teams, it will be broadcast to everybody.

6. Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for B followed by a run for C, but receive the response for C first.

   Do not request clarifications on when a response will be returned. If you have not received a response for a run within 30 minutes of submitting it, **you may have a runner ask the site judge to determine the cause of the delay. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.**

   If, due to unforeseen circumstances, judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams. This announcement will include a change to the 30 minute time period that teams are expected to wait before consulting the site judge.

7. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification. This includes submitting dozens of runs within a short time period (say, within a minute or two).

Your Programs

8. All solutions must read from standard input and write to standard output. In C this is `scanf()` / `printf()`, in C++ this is `cin` / `cout`, in Java this is `System.in` / `System.out`, and in Python this is `print()` and `input()`. The judges will ignore all output sent to standard error (`cerr` in C++, `System.err` in Java). You may wish to use standard error to output debugging information. From your workstation you may test your program with an input file by redirecting input from a file:

```
program < file.in
```

9. All lines of program input and output should end with a newline character (`\n`, `endl`, or `println()`).

10. All input sets used by the judges will follow the input format specification found in the problem description. You do not need to test for input that violates the input format specified in the problem.

11. Unless otherwise specified, all lines of program output should be left justified, with no leading blank spaces prior to the first non-blank character on that line.

12. Unless otherwise specified, all numbers in your output should begin with a '-' if negative, followed immediately by 1 or more decimal digits. If it is a real number, then the decimal point should be followed by as many decimal digits as can be printed. This means that for floating point values, use standard printing techniques (cout and System.out.println). Unless otherwise noted, the judging will check your programs with 10^{-3} accuracy, so only consider the sample output up until that point.

    In simpler terms, neither scientific notation nor commas will be used for numbers, and you should ensure you do not round or use a set precision unless otherwise specified in the problem statement.

13. If a problem specifies that an input is a floating point number, the input will be presented according to the rules stipulated above for output of real numbers, except that decimal points and the following digits may be omitted for numbers with no fractional component. Scientific notation will not be used in input sets unless a problem statement explicitly specifies it.

Good luck, and HAVE FUN!!!

A. Legion Roster

Emperor Aurelian has returned from yet another victory against the northern barbarians and wants to have a triumph, or victory parade, through the city. There are legions of soldiers coming from every province in the Empire to celebrate the victory. You are tasked with cooking for the entire army and therefore need to figure out how many soldiers there will be.

Given the number of legions taking part in the parade, how many soldiers will there be?

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

Each test case starts with a single integer, $0 < x \leq 200$, the number of legions in the parade. Each legion consists of 1,500 soldiers.

Output Format

Print out the number of soldiers in the parade for each test case.

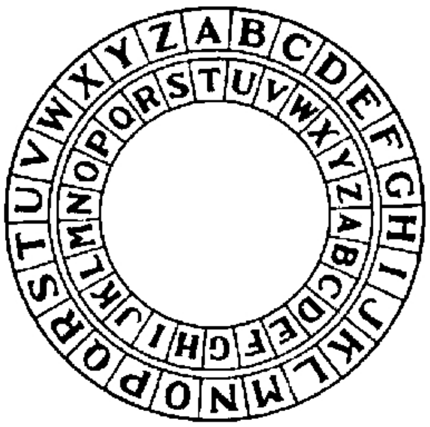
Sample Input

```
5
43
10
200
62
33
```

Sample Output

```
64500
15000
300000
93000
49500
```

B. Caesar's Cipher



Caesar has discovered an ingenious way to encode his messages. By shifting the value of each letter in his messages, he can make sure that Brutus will never discover his plans! Your job as Caesar's *scriba* is to encode Caesar's messages.

To encode a message, each lowercase character c is shifted by v, the shifting value, positions in the alphabet to produce the character s. For example, if $c =$ 'r' and $v = 4$, the shifted value will be $s =$ 't'. All letters in the messages will be lowercase, and overflow past the last letter of the alphabet 'z' cycles through the start of the alphabet. Thus, if $c =$ 'u' and $v = 8$, $s =$ 'c'.

Given a message and a shifting value, v, what will the encoded message be?

Input Format

The first line of the input will be a single integer $n \leq 10,000$. There will be n test cases that follow.

Each test case consists of two lines. On the first line will be the shifting value $0 \leq v \leq 128$, and on the second will be the plain message from Caesar. Each message will consist of only lowercase characters (a-z) and spaces.

Output Format

For each message, encode it using the provided shifting value and print it out on its own line.

Sample Input

```
4
2
et tu brute
5
carpe diem
18
ex nihilo nihil fit
100
audere est facere
```

Sample Output

```
gv vw dtwvg
hfwuj injr
wp fazadg fazad xal
wqzana aop bwyana
```

C. Target Practice

Everyone knows that the best archer of Mount Olympus is Diana, the Goddess of the Hunt. However, her twin brother Apollo disagrees, so he challenges her to an archery contest. Little does he know that the reason for Diana's skill is her understanding of physics that allows her to never miss.

Every shot in this competition is taken from the origin of a Cartesian plane towards a target at some position (x, y). The arrow is fired with velocity v_i m/s at an angle of θ radians above the +x axis.

The equations governing the flight of an arrow are:

$v_y = v_i * sin(\theta)$

$v_x = v_i * cos(\theta)$

$x = v_x * t$

$y = v_y * t - \frac{1}{2} * 9.8 * t^2$

Diana will always hit the target exactly. Given the x-position of the target (in m) find the height, or y-position, of her target.

Input Format

The input for this problem will begin with a single integer $n \leq 1,000$. There will be n test cases that follow.

Each test case will be on a single line consisting of three floating-point numbers. The first number is the initial velocity of the arrow $0 < v_i \leq 1,000$, in m/s. The second number is the angle at which Diana fires $-\frac{\pi}{2} < \theta < \frac{\pi}{2}$, in radians. The final number is the x-position of the target $0 < x \leq 10,000$ given in meters.

Output Format

For each test case output a single floating point number denoting the y-position of the target in meters. Your ouput value should be truncated to the third decimal place, and must be accurate to the thousandth place.

Sample Input

```
2
1 0 1
10 0.78539816339 2
```

Sample Output

```
-4.9
1.607
```

D. Gods Abound

Jupiter, Janus, and Juno; Mars, Mercury, and Minerva; Vulcan, Venus, and Vesta. The Romans had many, many gods. And while they're capable enough to remember the names, they need a bit of help remembering that Mercury was the god of blacksmithing, or was it Vulcan? Either way, your task is to write a program to help keep track of which god does what.

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

The first line of each test case will contain two integers, the number of gods $0 < g \leq 1,000$ and the number of queries $q < 1,000$. g lines will then follow, each with the name of a god and the role of that god, for example, "Vulcan Blacksmithing". Each godly name and role will consist only of letters and no spaces, but the name and the role will be separated by a space. No god will be listed twice. q queries will follow, one on each line, with the name of a god, such as "Vulcan".

Output Format

Output the corresponding role of the god in each query on its own line.

Sample Input

```
1
4 3
Saturn Sky
Vulcan Blacksmithing
Mercury Speed
Janus Choices
Mercury
Vulcan
Janus
```

Sample Output

```
Speed
Blacksmithing
Choices
```

E. Fortune Teller of Fraud?

The oracle Pythia claims she can predict the future. A skeptic wants to prove she is a fraud, so they agree to a challenge. Pythia will predict the temperature of the next 10 days. 10 days later, the skeptic will return with the actual daily temperatures. If she gets at least 8 correct, the skeptic will accept her as the Oracle. If she gets fewer than 5 correct, the skeptic will declare her a fraud. If she gets 5,6, or 7 correct it is a draw. Given Pythia's predictions and the actual temperatures, determine the belief of the skeptic.

Input Format

Input will begin with a single integer, $n \leq 1,000$. There will be n test cases that follow. Each test case will consist of two lines. The first line will consist of Pythia's prediction as 10 space-separated integers $0 \leq x_i \leq 50$. The second line will be the actual temperatures, also as 10 space-separated integers $0 \leq y_i \leq 50$.

Output Format

For each case, print "Oracle" if the skeptic believes Pythia, "Fraud" if the skeptic declares Pythia so, or "Draw" if the results are inconclusive. Each output should be separated by line.

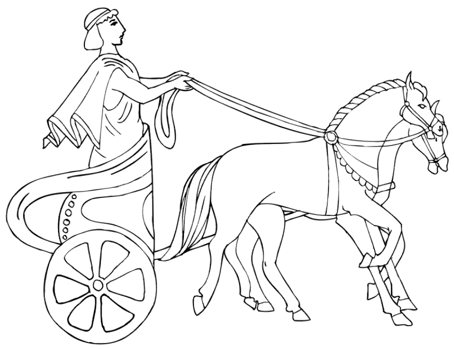
Sample Input

```
3
49 7 10 3 39 41 9 27 9 48
49 7 9 3 39 41 9 27 9 48
32 33 31 22 18 28 37 38 3 14
18 33 31 22 20 28 37 40 3 10
7 4 25 38 5 16 4 22 37 38
4 2 25 35 3 19 5 22 37 38
```

Sample Output

```
Oracle
Draw
Fraud
```

F. Circus Sorting



All hail the greatest sport ever to exist: Roman chariot racing! Last week, the biggest race of the year was run for the Roman Games at the Circus Maximus. But somehow in all the commotion, the ranking sheet was lost! The officials still have the list of each chariot's time, however. Can you give them back the list of the rankings?

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

The first line of each test case will consist of a single integer, $c < 10,000$, the number of competitors. c lines will follow, each with a string m, the alphanumeric name of a competitor, and $t < 10,000$, a floating-point time value of when the competitor finished. m and t are separated by a single space character.

Output Format

Output the list of the competitors' names, one per line, in the order of increasing corresponding time.

Sample Input

```
1
6
achilleus 47.5
cronus 34.67
cyrus 501.42
septimus 761.55
lucius 94.91
magnus 12.1
```

Sample Output

```
magnus
cronus
achilleus
lucius
cyrus
septimus
```

G. Pesky Publicans

Tax collectors in the Roman empire, called *publicans*, are not well perceived, and for good reason. However, they are necessary for the running of the empire. Your job is to estimate how much money Rome will receive in taxes. Not everyone pays, but at least one person is guaranteed to pay. The trouble is you can never really know who will and won't pay.

Given a list of what citizens would pay if they did pay taxes, output the average of all possible sums of taxes received.

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

Each test case begins with a single integer p denoting the number of people in Rome, $1 \leq p \leq 10,000$. The next line will contain p space separated integers, $0 < v_i < 10,000$, denoting the amount of taxes that each citizen would pay if they decided to pay.

Output Format

Each test case should contain a single floating point number denoting the average taxes paid, accurate and truncated to three decimal places.

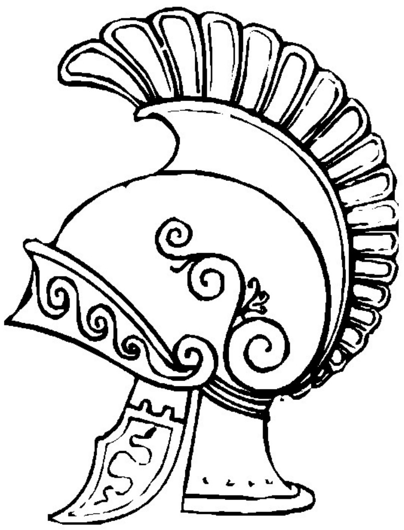
Sample Input

```
2
5
1 2 3 4 5
5
10 11 12 13 14
```

Sample Output

```
7.742
30.967
```

H. Et Tu Brute?



Caesar has been betrayed! He always thought his best friends were the senators of Rome, but it turns out they are trying to kill him. Even his best friend, Marcus Junius Brutus, is in on the plot. Luckily, he's managed to get away, for now. If he can make it out of the senate he'll be okay, and he can overthrow the senate and take over Rome.

Unfortunately, he's a little rattled, and is not making smart decisions. Every minute, he is randomly choosing which room he enters and hides in for the next minute. In each room there is a $p\%$ chance that he will be captured. After m minutes the senators will give up the search and go back to their homes, allowing him to escape. Calculate the chance that he will escape if he chooses rooms at random.

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

Each test case will start with a line of 2 integers: the number of rooms $5 \leq r \leq 50$ and the number of minutes Caesar must survive, $m \leq 10$. The next r lines will consist of a floating point number $p < 1$ denoting the probability that Caesar survives for the next minute in this room and 4 integers between 1 and r (inclusive on both ends) denoting which rooms the current room is connected to. Caesar always spends the first minute in room 1.

Note: The senate is weird, just because room a leads to room b does not mean Caesar can get back into room a from room b.

Output Format

Output a single floating point number denoting the probability that Caesar survives. The output should be accurate and truncate to three decimal places.

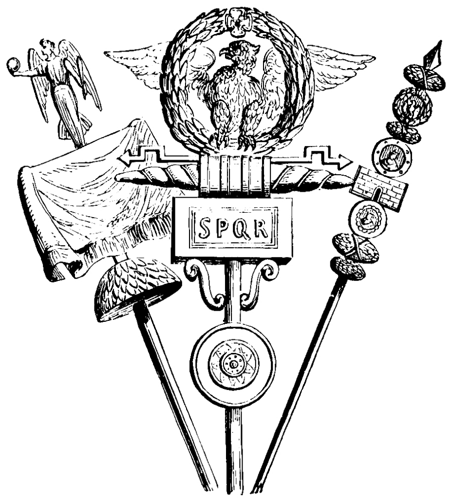
Sample Input

```
1
5 3
0.50 2 3 4 5
0.25 1 3 4 5
0.30 1 2 4 5
0.80 1 2 3 5
0.60 1 2 3 4
```

Sample Output

```
0.113
```

I. Marching Legion



There's a rebellion in Antioch! Caesar is preparing to dispatch an entire legion of his best troops to quell the rebellion, but before he does, he wants to make certain the legion will make it to Antioch. Along the way, there are mountains, rivers, seas, and mythical monsters, each of which the legion cannot pass through. Given a map a legion must travel through, find out if the legion can reach Antioch from Rome.

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

Input will begin with two integers, $0 < h, w < 100$, indicating the height and width of the map, respectively. h lines with w characters each will follow. An "X" on the map represents an impassable obstacle while an "O" represents traversable terrain. Note that legions, moving in square formations, can only move up, down, left, and right; legions can never travel diagonally. "R" represents Rome, the starting point of the legion, and "A" represents Antioch, the destination point for the legion.

Output Format

Output "March onward!" if the legion can reach Antioch from Rome, "Stay home!" otherwise. Each output should be line separated.

Sample Input

```
3
6 6
XXXXOA
XOXXOX
XOXXOX
XOOOOX
XOXXXX
XROOOX
4 4
XXOA
XOOX
XOXX
XRXO
2 2
XA
RX
```

Sample Output

```
March onward!
March onward!
Stay home!
```

J. Convivium Combat

Caesar has just returned victorious from conquering a far-off land, and it's celebration time. The senate has thrown a great feast in Caesar's honor, and no Roman *convivium* is complete without a food fight. The Romans were nothing if not orderly in their fighting, and the same held for their food fights. At the start of the food fight, the cuisine combatants would divide in to two even groups. Then, each member of each group would pick a target in the other group who they will barrage with their meal. Caesar wants to win this food fight, and he knows that the initial barrage matters the most. If each of his food fighters can pick exactly one target, how many members of the other team can be hit in the initial barrage?

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

The first line of each test case will consist of a single integer $0 < c \leq 5,000$, indicating the number of cuisine combatants on Caesar's team. c lines will follow. Each line will have the food fighter's name, the number of targets they can hit, $0 \leq t \leq n$, and the names of their t targets, each separated by a space. Members of Caesar's team may share names with members of the opposing team, but no two members of the same team will share a name. All names will be alphanumeric and not have spaces.

Output Format

Output the maximum number of distinct targets Caesar's team can hit with their initial food barrage.

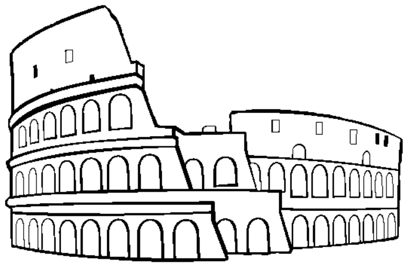
Sample Input

```
1
10
Saltius 3 Aufeius Ulpius Fulginas
Ragonius 3 Maenius Sidonius Julius
Vicirius 3 Pinarius Maenius Ulpius
Septimius 1 Pinarius
Cocceius 1 Sidonius
Trebius 4 Julius Dellius Sabucius Pinarius
Cossinius 0
Aelius 4 Maenius Sabucius Ulpius Aufeius
Catius 3 Julius Fulginas Dellius
Vitellius 2 Julius Sabucius
```

Sample Output

```
9
```

K. Casino Romale



It's a showdown in the Coliseum! The Roman Empire's finest warriors have flocked to the city of Rome for the world's first official rock, paper, scissors tournament. The excitement is palpable; even the townsfolk and farmers are getting into it! The emperor himself has said that he will play against the winner of the tournament, and if the emperor loses, he will pay the winner 1 million silver *denarius*.

While this is great fun for nearly everyone involved, the treasurer isn't so happy about it. Not only is it a needless expense, it's also uncertain. To make his life a bit easier, why don't you find the expected value of the pay out, rounded to the nearest silver denarius? For example, if the winner of the tournament would beat the emperor 9 times out of 10, then the expected pay out is 900,000 silver denarii.

To make your job easier, scouters have determined the probability that each contestant will play rock, paper, or scissors. For example, if a given contestant's chances are 0.25, 0.5, and 0.25 respectively, then each round, they may choose either rock, paper, or scissors, but they are twice as likely to pick rock as paper or scissors.

The tournament works as follows. Contestant 1 first goes against Contestant 2, Contestant 3 goes against Contestant 4, and so on. Then the winner of 1 vs 2 moves on to go against the winner of 3 vs 4, winner of 5 vs 6 moves on to go against the winner of 7 vs 8, and so on. Then the winner of 1, 2, 3, and 4 goes against the winner of 5, 6, 7, and 8. It continues like this until there is only one contestant remaining, who then goes against Caesar.

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

The first line of each test case will be the number of contestants, $c = 2^m$, where $m \leq 16$. c lines will follow, each with three floating-point values, $0 < r_i, p_i, s_i \leq 1$, $r_i + p_i + s_i = 1$ indicating the probabilities that contestant i will pick rock, paper, or scissors in each match respectively. One more line will follow consisting of three floating-point values $0 \leq r_c, p_c, s_c \leq 1$, $r_c + p_c + s_c = 1$, indicating the probability Caesar will pick rock, paper, or scissors.

Output Format

Output the number of silver *denarius* Caesar can expect to pay the winner of the tournament. For example, if Caesar has a 50% chance of losing, output would be "500000".

Sample Input

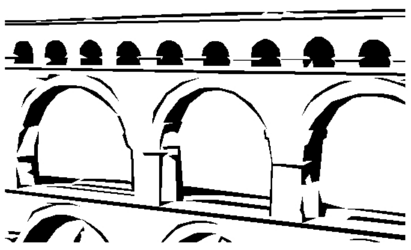
```
1
4
0.4 0.3 0.3
0.2 0.1 0.7
0.1 0.4 0.5
```

```
0.4 0.4 0.2
0.2 0.6 0.2
```

Sample Output

```
544399
```

L. Aqueduct Rider



Water parks can be a lot of fun, right? The Roman children sure would have thought so! Unfortunately, they didn't have any proper water parks, so instead they used aqueducts.

Aqueducts make long, interweaving paths all throughout Rome. Since aqueducts are constantly splitting and combining, there are millions of possible ways to get to the bottom of an aqueduct. The path they take can be drastically different depending on which branches they slide down. While the number of paths down is numerous, it is not infinite; once you go down a portion of the aqueduct, you can never go back up and there is no path to get to the start of that segment.

Your task is to help the Roman hero Hercules in finding the longest path down an aqueduct network so he can tell the children of Rome.

Input Format

The first line of the input will be a single integer, $n \leq 1,000$. There will be n test cases that follow.

The first line of each test case consists of two integers $0 < x, y \leq 100,000$, the number of sections in the aqueduct network. x lines will follow, each with four integers i, u, v, and l such that $0 \leq i, u, v \leq 100,000, v - u \leq l \leq 100,000$, indicating the ID, starting height, ending height, and length of each aqueduct section, respectively. y more lines will follow, each with two integers $0 \leq h, k \leq 100,000$, indicating that the aqueduct sections with IDs h and k are connected.

Output Format

For each test case, output the length of the longest path down the aqueduct network as a single integer. Each output should be separated by line.

Sample Input

```
1
9 13
0 8 4 6
5 15 8 11
4 15 8 8
2 8 0 14
7 15 4 14
3 15 8 12
1 8 4 4
6 15 4 11
8 4 0 7
5 2
4 1
3 1
0 8
4 0
```

```
5 0
6 8
1 8
3 2
4 2
5 1
3 0
7 8
```

Sample Output

```
26
```