0 DIE with a T

I am on a diet. So I can only eat upto 2,000 kcal (2,000,000 cal) a day. I eat 4 meals a day(*b*reakfast, *l*unch, *s*nack, and *d*inner) and life would be easy if every food was exactly 500,000cal. But it isn't. So help me out. From 4 different groups of foods(each group representing the 4 meals I eat) each consisting of N foods, compute whether it is possible to select a food from each group so that the total calorie intake is *exactly* 2,000,000 cal.

INPUT: The first line is integer N. The *i*-th line of next N lines contains the four calorie value of each meal, b_i, l_i, s_i, d_i.

OUTPUT: Print **SUCCESS** if we can choose exactly 4 food so that the total of calorie equals 2,000,000. ($b_i + l_j + s_k + d_l = 2,000,000$) Print **FAIL** elsewise.

CONDITION: $4 \leq N \leq 10,000$, $0 \leq b, l, s, d \leq 1,000,000$

EXAMPLE:

```
2
100000 600000 400000 600000
200000 500000 600000 800000
```

```
SUCCESS
```

```
1
100000 600000 400000 600000
```

```
FAIL
```

HINT: The obvious approach would be to try every combination of 4 foods from each meal. However, this approach takes time proportional to N^4, which is 10000000000000000 in the worst case. Therefore, instead of trying every combination, we need a better approach. One possible approach would be computing *partial* calorie k in time proportional to N^2, and checking whether $(2000000 - k)$ exists in $constant$ time. This can be done by a set. Using a set also has the added benefit of removing duplicates, reducing computation time somewhat.

1 Statistics Showdown

Given $N (N\%2 == 1)$ numbers, compute the following representative values.

1. Arithmetic Mean
2. Geometric Mean
3. Median
4. Mode (The most common value)
5. Range $(max(n_i) - min(n_i))$

INPUT: The first line is integer N. The *i*-th line of next N lines contains a single integer n_i.

OUTPUT: Print 5 lines, each consisting of the Arithmetic Mean, Geometric Mean, Median, Mode, and Range of n_i. If there are more than 1 mode, print any of those. For Arithmetic and Geometric mean, round to one decimal place (*e.g.* $1.75 = 1.8, 3.74 = 3.7$).

CONDITION: $1 \leq N \leq 200$ (N is odd number), $0 \leq n_i \leq 1000$

EXAMPLE:

```
5
1
4
2
3
5
```

```
3.0
2.6
3
1
4
```

HINT: Googling "root in python (other than square root)" would help calculating the Geometric Mean.

---

2 Prefix Calculator

We usually write mathematical expressions this way: $3 + 4 * 2$. This is called an infix notation, and although easy to understand, it has some drawbacks. Therefore, there are other notations like prefix notations(operators precede their operands), or postfix notations(operators follow their operands). For example, using prefix notation, the expression above becomes $(+\ 3\ (*\ 4\ 2))$. To evaluate this expression, we start from the left, and when we see an operator('+'), we expect two sub-expressions. We see a 3 and (* 4 2). 3 is 3. For (* 4 2), we repeat the step above, with an operator('*') and two sub-expressions (4 and 2). Since there are no more sub-expressions to be evaluated, we calculate $(*\ 4\ 2) = 8$, and $(+\ 3\ 8) = 11$.

TLDR; we are going to implement a mathematical expression calculator. For your convenience, I have already implemented the $infix\_prefix(exp)$ (There is no need to understand how it works). It's your job fill in $calculate(exp)$ that evaluates the prefix notation, *recursively*.

INPUT: A single line of a correct mathematical expression, with parentheses.

OUTPUT: The result of calculating the mathematical expression.

CONDITION: For simplicity, the numbers used in the expression is a single digit number, ranging from 0 to 9. The operators used are $+(addition)$, $-(substraction)$, $*(multiplication)$, $/(division)$.

EXAMPLE:

```
(3+2)*2
10
```

HINT: Before implementing calculate(prefix_exp), examine the prefix notation of expressions. We can see that the prefix expression is formed as following:

$$\begin{array}{rll} prefix\_exp ::= & (operand,\ prefix\_exp,\ prefix\_exp) & tuple \\ | & n & integer \end{array}$$

When the type of the prefix_exp is integer, (in Python: $isinstance(prefix\_exp,\ int)$), there is nothing to do; return n. This becomes the basecase of recursion. When it's a tuple, we have to do some calculation. Assuming that $calculate(prefix\_exp)$ will correctly calculate the results of the two subexpression would help.

---

3 Heart Signal

It is not often the case that someone you love, loves you back. It was easily observed in the heartwarming(or breaking?) tvshow Heart Signal. Now, as a fan of the show, it is of our interest to see if the participants can fulfill their love. Meaning that, someone who the participant likes also likes the participant back.

INPUT: The first line is integer N. The i-th line of next N lines contains the participant's name, and the someone he/she likes, seperated by a space. The N+2-th(last) line contains the name of a participant.

OUTPUT: If the participant is also liked by the someone he/she likes, print **LOVE**. print OUCH elsewise.

EXAMPLE:

```
5
hulk blackwidow
hawkeye blackwidow
blackwidow ironman
ironman pepper
pepper ironman
hulk

OUCH
```

```
6
hulk blackwidow
hawkeye blackwidow
blackwidow ironman
ironman pepper
pepper ironman
pepper

LOVE
```

HINT: We would be using a list if the names were just integers...