1 Problem Statement

1.1 Random number generator

There is an ideal random number generator, which given a positive integer M can generate any real number between 0 to M, and probability density function is uniform in [0, M].

Suppose we generate 2 numbers x and y via the generator by giving it 2 positive integers A and B, what's the probability that x + y is less than C? where C is a positive integer.

1.1.1 Input Format

The first line of the input is an integer N, the number of test cases.

N lines follow. Each line contains 3 positive integers A, B and C.

1.1.2 Constraints

All the integers are no larger than 10000.

1.1.3 Output Format

For each output, output a fraction that indicates the probability. The greatest common divisor of each pair of numerator and denominator should be 1.

1.1.4 Sample Input

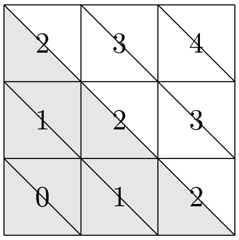
```
3
1 1 1
1 1 2
1 1 3
```

1.1.5 Sample Output

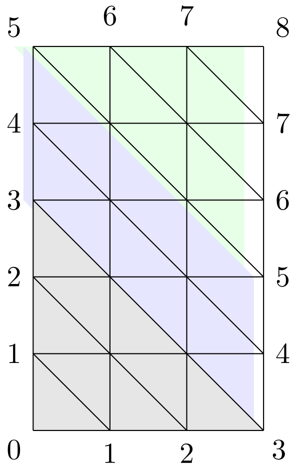
```
1/2
1/1
1/1
```

2 The Solution

For the case, where a, b and c are equal, we see that the answer is $1/2$.



For the $3, 4, 5$ case you have the following solutions: $\frac{1}{30}$ $\frac{2}{15}$ $\frac{3}{10}$ $\frac{1}{2}$ $\frac{7}{10}$ $\frac{13}{15}$ $\frac{29}{30}$ $\frac{1}{1}$.



$$r = \begin{cases} \frac{c^2}{2ab} & \text{if } c \leq a; \\ \frac{2ac-a^2}{2ab} & \text{if } c \geq a \wedge c \leq b; \\ \frac{2(ac+bc)-a^2-b^2-c^2}{2ab} & \text{if } c \geq b \wedge c \leq a+b \\ 1 & \text{if } c \geq a+b. \end{cases}$$

$\langle algo \rangle \equiv$

```
int numerator = 2 * a * b;
if (a + b <= c) {
  numerator = 2 * a * b;
} else
if (c < a) {
  numerator = c * c;
} else
if (c < b) {
  numerator = 2 * a * c - a * a;
} else {
  numerator = 2 * a * b - (a + b - c) * (a + b - c);
}

output_reduced(numerator, 2 * a * b);
```

There are 2 conditions that we must test on input. We must make sure that a, b and c are positive integers.

⟨*input checking*⟩≡

```
if (a * b * c == 0) {
  fprintf(stderr, "a, b and c must be positive integers!\n");
  return(-1);
}
```

And we must make sure that a is less than b. We do this by swapping a with b if it is not the case.

⟨*input checking*⟩+≡

```
if (a > b) {
  a ^= b;
  b ^= a;
  a ^= b;
}
```

The output is a reduced normalized fraction. We need a *greatest common divisor* funtions to reduce the fraction.

⟨*gcd*⟩≡

```
unsigned int gcd(unsigned int u, unsigned int v) {
  while ( v != 0) {
    unsigned int r = u % v;
    u = v;
    v = r;
  }
  return u;
}
```

Then we have a procedure prints a fraction to standard output.

⟨*output reduced fraction*⟩≡

```
void output_reduced(int n, int d)
{
  unsigned int div = gcd(n,d);
  int x = n / div;
  int y = d / div;
  printf("%d/%d\n", x, y);
}
```

The *main*, entry point, function reads the input as described in section 1.1.1.

⟨*main function*⟩≡

```
main()
{
  int lines;
  if (scanf( "%d\n", &lines)) {
    for (int i = 0; i < lines; i++) {
      unsigned int a, b, c;
      if (scanf( "%u %u %u\n", &a, &b, &c)) {
        ⟨input checking⟩
        ⟨algo⟩
      }
    }
  }
  return 0;
}
```

⟨*⟩≡

```
  #include <cstdio>

  using namespace std;

  ⟨gcd⟩

  ⟨output reduced fraction⟩

  ⟨main function⟩
```