Árboles y Grafos
Tarea 2

Este es el enunciado de la Tarea 2 del curso *Árboles y Grafos*, 2024-2. La tarea está compuesta por una parte teórica y una parte práctica. La parte teórica de la tarea se puede realizar en **parejas** o de forma **individual**. La parte práctica debe realizarse de forma **individual**. Sus soluciones deben ser entregadaa más tardar el día 18 de Agosto a las 23:59. Para la parte téorica se debe entregar un documento llamada `tarea2.pdf`. Para la parte práctica las entregas se deben hacer a través de la arena de programación. En caso de dudas y aclaraciones puede escribir por el canal `#tareas` en el servidor de Discord del curso o comunicarse directamente con el profesor y/o el monitor.

Invariantes de ciclo y complejidad computacional [50 pts.]

1. Considere el siguiente algoritmo:

```python
def algoritmo1(N):
  ans, ac, i = [], 1, 1
  while i <= N:
    ans.append(ac)
    ac = ac * (i + 1)
    i = i + 1
  return ans
```

   Indique qué cálcula este algoritmo y realice el análisis de la correctitud del mismo siguiendo los pasos vistos en clase.

2. Considere el siguiente algoritmo:

```cpp
vector<int> algoritmo2(int N){
  vector<int> ans;
  int i = 2;
  while(N > 1){
    if(N % i == 0){
      ans.push_back(i);
      N = N / i;
    }
    else
      i += 1;
  }
  return ans;
}
```

   Indique qué cálcula este algoritmo y realice el análisis de la correctitud del mismo siguiendo los pasos vistos en clase.

3. Considere el siguiente algoritmo:

```python
def algoritmo3(l):
  i = 0
  while i < len(l):
    j, tmp, pos = i + 1, l[i], i
    while j < len(l):
      if l[j] < tmp:
        tmp = l[j]
        pos = j
```

```
9         j += 1
10      l[i], l[pos] = tmp, l[i]
11      i += 1
```

Indique qué cálcula este algoritmo y realice el análisis de la correctitud del mismo siguiendo los pasos vistos en clase.

4. Demuestre o refute las siguientes afirmaciones:

   a) $6n^2 + 18n \in O(4n^2 log\ n)$

   b) $2^{2n} \in O(2^n)$

   c) $2^{n+2} \in O(2^n)$

Programación con Dividir y Conquistar [50 pts.]

5. Hay tres problemas prácticos cuyos enunciados aparecen a partir de la siguiente página.

A - Problem A

Source file name: `bids.cpp`
Time limit: 1 second

Deciding the price of a product is not an easy matter. If the price is high, then consumers get angry because they think it is too expensive. But if the price is low, then producers get angry because they do not earn enough money.

In this problem, you have to bid for the price that produces the minimum number of angry people.

For a given product, we have a set of producers and a set of consumers. Each producer has suggested his/her ideal price for the product. If your bid is below that ideal price, then the producer will get angry. Also, each consumer has suggested his/her ideal price for the product. If your bid is above that ideal price, then the consumer will get angry.

You have to find the price that produces the minimum number of angry people (either producers or consumers). Your bid price should always be between 0 and 10^9. And if there is more than one optimal solution, you have to output the lowest one.

Input

The input consists of a series of test cases. The first line contains a number that indicates the number of test cases.

Each test case consists of three lines. The first one contains two integers, P and C, indicating the number of producers and consumers of the product, respectively. These numbers will be between 0 and 10^4. Then, the second line contains P integer numbers, the ideal prices for the producers. And the third line contains C integer numbers, the ideal prices for the consumers. The prices are between 1 and 10^8.

The input must be read from standard input.

Output

For each input case, you have to output two numbers separated by one space: the bid price that produces the minimum number of angry people; and the total number of angry people for that price. Remember that if there is more than one optimal solution, you have to output the one with lowest price; and the result cannot be a negative number.

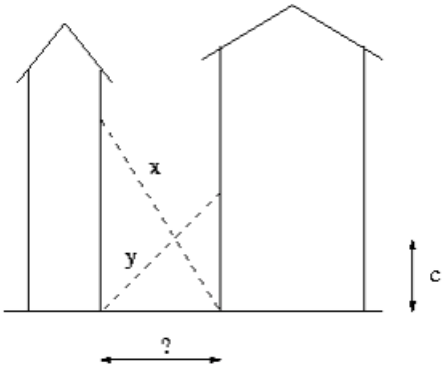
The output must be written to standard output.

Sample Input	**Sample Output**
4 3 3 10 16 12 21 20 25 4 3 104 120 98 132 120 88 140 8 8 36 27 52 72 36 37 26 38 35 75 36 39 44 82 23 62 0 2 28 71	16 0 120 2 38 5 0 0

B - Problem B

Source file name: `ladders.cpp`
Time limit: 1 second

A narrow street is lined with tall buildings. An x foot long ladder is rested at the base of the building on the right side of the street and leans on the building on the left side. A y foot long ladder is rested at the base of the building on the left side of the street and leans on the building on the right side. The pointvwhere the two ladders cross is exactly c feet from the ground. How wide is the street?



Input

Each line of input contains three positive floating point numbers giving the values of x, y, and c.

The input must be read from standard input.

Output

For each line of input, output one line with a floating point number giving the width of the street in feet, with three decimal digits in the fraction.

The output must be written to standard output.

Sample Input	**Sample Output**

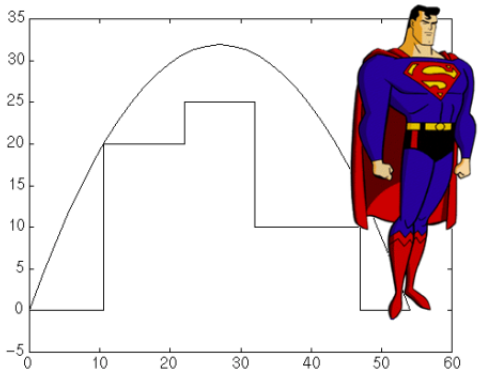
`30 40 10`
`12.619429 8.163332 3`
`10 10 3`
`10 10 1`
	`26.033`
	`7.000`
	`8.000`
	`9.798`

C - Problem C

Source file name: `leaps.cpp`
Time limit: 1 second

It's a bird! It's a plane! It's coming right at us!

Although it sometimes seems like it, Superman can't fly (without a plane). Instead, he makes super-human leaps, especially over tall buildings. Since he never knows when he will need to catch a criminal, he can't register flight paths. To avoid hitting planes, he tries to keep his jumps as low to the ground as he can. Given a city-scape as input, find the angle and velocity of Superman's jump that minimizes his maximum altitude.



Recall that gravity provides an acceleration of $9.8m/s^2$ downwards and the formula for Superman's vertical distance from his starting location is $d(t) = vt + 0.5at^2$ where v is his initial velocity, a is his acceleration and t is time in seconds since the start of the leap.

Input

Input consists of a sequence of city-scapes, each of the form

n

$0\ d_1$

$h_2\ d_2$

$\vdots$

$h_{n-1}\ d_n - 1$

$0\ d_n$

Superman starts at ground level and leaps $d_1 + \ldots + d_n$ metres, landing at ground level and clearing all of the buildings at heights h_2 to h_{n-1}, each with the given widths. n will be at most 100.

The input must be read from standard input.

Output

Output is the angle and initial velocity that minimizes the height that Superman attains, both appearing on the same line. The values should be given to two decimal places and be accurate within 10^{-5} degrees or m/s, as appropriate.

The output must be written to standard output.

Sample Input	**Sample Output**
3 0 5 10 5 0 5 5 0 10.5 20 11.5 25 10 10 15 0 7	71.57 15.65 67.07 27.16