Este es el enunciado de la Tarea 2 del curso *Árboles y Grafos*, 2025-2. 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 entregadas a más tardar el día 25 de Agosto a las 23:59. Para la parte teórica se debe entregar un documento llamado `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 la monitora.

Invariantes de ciclo y complejidad computacional [50 pts.]

1. Considere el siguiente algoritmo:

```python
def algoritmo1(A, v):
  i = 0
  ans = 0
  while i < len(A):
    if A[i] % v == 0:
      ans = ans + A[i]
    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:

```python
def algoritmo2(N, v):
  i = 0
  ac = 1
  ans = []
  while i <= N:
    ac = ac * v
    ans.append(ac)
    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(A):
  i = 1
  ans = 0
  tst = A[0]
  while i < len(A):
    if A[i] >= tst:
      tst = A[i]
      ans = 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.

4. Demuestre o refute las siguientes afirmaciones:

   a) $5n^2 - 2n \in O(4n^2 \log n)$

   b) $8^n \in O(4^n)$

   c) $2 \log_2 n \in O(\log_8 n)$

Programación con Dividir y Conquistar [50 pts.]

5. Hay dos problemas prácticos cuyos enunciados aparecen a partir de la siguiente página. Es necesario incluir en la cabecera de cada archivo el análisis de la complejidad de la solución en comentarios.

A - Problem A

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

Rainfall is measured in millimeters. The rain is collected in a vertical transparent tube with millimeter markings, and once the rain has stopped falling, one can check the height of the water in the tube.

In our problem, the tube unfortunately has a leak at height L millimeters (mm). If the water level is above the leak then water drains from the tube at a rate of K millimeters per hour (mm/h).

We want to figure out how much rain fell during a particular rainfall. We assume that the tube is high enough that it does not overflow. We also assume that rain falls at an (unknown) uniform rate during a rainfall, and that water does not evaporate from the tube. The height of the leak itself is also negligible.

Input

The first line of the input file contains an integer N which denotes the total number of test cases. The description of each test case is given next. A line with five blank-separated positive numbers L, K, T_1, T_2, and H, where:

- L is where the leak is (mm)
- K is the rate at which water leaks (mm/h)
- T_1 is the duration of the rainfall (h)
- T_2 is the time between the end of the rainfall and the observation of the water level (h)
- H is the water level in the tube when we observe it (mm)

Each number is at least 0.01 and at most 1000.00, and each is given with two decimals.

The input must be read from standard input.

Output

For each test case print one line with two blank-separated floating point numbers F_1 and F_2, where F_1 is the smallest rainfall in millimeters that would result in the given observation and F_2 is the largest rainfall in millimeters that would result in the given observation. Values need to be printed with 4 decimals.absolute or relative error smaller than 10^{-7}.

The output must be written to standard output.

Sample Input	Sample Output

2
80.00 0.50 2.00 1.50 80.00
150.00 1.00 100.00 150.00 100.00
	80.0000 80.7594
	100.0000 100.0000

B - Problem B

Source file name: `souvenirs.cpp`
Time limit: 3 seconds

During one of his rare "rest" trips — because *Zlatan never gets tired*, he just decides when to shine — Zlatan visited an exclusive souvenir shop in Stockholm to buy gifts for his loyal fans.

But this was no ordinary shop: the pricing rules were so strange that even Zlatan raised an eyebrow. There are n different souvenirs, numbered from 0 to $n-1$. The i-th souvenir has a base price of a_i Swedish kronor.

In this shop, if Zlatan buys k souvenirs with indices $x_0, x_1, \ldots, x_{k-1}$, the cost of souvenir x_j is:

$$a_{x_j} + (x_j + 1) \cdot k$$

The shopkeeper claims "the index reflects the prestige of the souvenir". Zlatan didn't quite understand, but smiled — and everyone pretended the rule made sense. In other words, the final cost of each souvenir is equal to its base price plus its index multiplied by the total number of souvenirs Zlatan buys.

Zlatan wants to take home as many souvenirs as possible without spending more than S Swedish kronor. He cannot buy the same souvenir twice — because Zlatan never repeats gifts. . . unless the gift is himself. If there are multiple ways to get the maximum number of souvenirs, Zlatan will choose the one with the minimum total cost — even though money is not a problem for him, it's just a matter of style.

Your task is to help Zlatan decide what to buy.

Input

For each test case there are two lines as follows:

- The first line contains two integers n and S ($1 \leq n \leq 10^5$, $1 \leq S \leq 10^9$) — the number of souvenirs and Zlatan's budget in kronor.
- The second line contains n integers $a_0, a_1, \ldots, a_{n-1}$ ($1 \leq a_i \leq 10^9$) — the base prices of the souvenirs.

The input must be read from standard input.

Output

For each case print two integers:

- The maximum number of souvenirs Zlatan can buy.
- The minimum total cost to buy that many souvenirs.

Note: In the first test case, Zlatan buys souvenirs 0 and 1:

Souvenir 0: $2 + (0 + 1) \cdot 2 = 4$

Souvenir 1: $3 + (1 + 1) \cdot 2 = 7$

Total cost = $4 + 7 = 11$.

The output must be written to standard output.

Sample Input	Sample Output

3 11
2 3 5
5 20
1 10 3 2 5
	2 11
	2 12