Este es el enunciado de la Tarea 2 del curso *Árboles y Grafos*, 2025-1. 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 entregada a más tardar el día 16 de Febrero a las 23:59. Para la parte téorica 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(N, v):
  ac = 1
  ans = []
  i = 0
  while i <= N:
    ans.append(ac)
    ac = ac * v
    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(lval, lpos):
  i = 0
  j = 0
  ans = 0
  while i < len(lpos):
    if j < lpos[i]:
      j += 1
    else:
      ans = ans * 10 + lval[j]
      j += 1
      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):
  m = None
  r = 0
  j = len(A) - 1
  while j >= 0:
    i, ac = 0, 0
    while i < j:
```

```
8       if A[i] <= A[j]:
9         ac = ac + 1
10      i = i + 1
11    if ac > r:
12      ac = r
13      m = A[j]
14    j = j - 1
15  return m
```

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) $7n^2 - 3n \in O(3n^2 sqrtn)$

   b) $2^n \in O(n!)$

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

Programación con Dividir y Conquistar [50 pts.]

5. Hay tres 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: `bit.cpp`
Time limit: x seconds

The bitmap is a data structure that arises in many areas of computing. In the area of graphics, for example, a bitmap can represent an image by having a `'1'` represent a black pixel and a `'0'` represent a white pixel.

Consider the following two ways of representing a rectangular bit map. In the first, it is simply represented as a two dimensional array of 1's and `0`'s. The second is based on a decomposition technique. First, the entire bit map is considered. If all bits within it are 1, a `'1'` is output. If all bits within it are 0, a `'0'` is output. Otherwise, a `D` is output, the bit map is divided into quarters (as described below), and each of those is processed in the same way as the original bit map. The quarters are processed in top left, top right, bottom left, bottom right order. Where a bit map being divided has an even number of rows and an even number of columns, all quarters have the same dimensions. Where the number of columns is odd, the left quarters have one more column than the right. Where the number of rows is odd the top quarters have one more row than the bottom. Note that if a region having only one row or one column is divided then two halves result, with the top half processed before the bottom where a single column is divided, and the left half before the right if a single row is divided.

Write a program that will read in bitmaps of either form and transform them to the other form.

Input

Input will consist of a series of bit maps. Each bit map begins with a line giving its format (`'B'` or `'D'`) and its dimensions (rows and columns). Neither dimension will be greater than 200. There will be at least one space between each of the items of information. Following this line will be one or more lines containing the sequence of `'1'`, `'0'` and `'D'` characters that represent the bit map, with no intervening spaces. Each line (except the last, which may be shorter) will contain 50 characters. A `'B'` type bitmap will be written left to right, top to bottom.

The file will be terminated by a line consisting of a single `'#'`.

The input must be read from standard input.

Output

Output will consist of a series of bitmaps, one for each bit map of the input. Output of each bit map begins on a new line and will be in the same format as the input. The width and height are to be output right justified in fields of width four.

The output must be written to standard output.

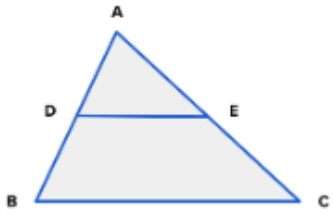
Sample Input	**Sample Output**

`B 3 4`
`001000011011`
`D 2 3`
`DD10111`
`#`
	`D 3 4`
	`D0D1001D101`
	`B 2 3`
	`101111`

B - Problem B

Source file name: `triangle.cpp`
Time limit: x seconds

See the picture below.



You are given **AB**, **AC** and **BC**. **DE** is parallel to **BC**. You are also given the area ratio between **ADE** and **BDEC**. You have to find the value of **AD**.

Input

Input starts with an integer T, denoting the number of test cases.

Each case begins with four real numbers denoting **AB**, **AC**, **BC** and the ratio of **ADE** and **BDEC** (**ADE**/**BDEC**). You can safely assume that the given triangle is a valid triangle with positive area.

The input must be read from standard input.

Output

For each case of input you have to print the case number and **AD**. Errors less than 10-6 will be ignored.

The output must be written to standard output.

Sample Input	**Sample Output**

`4`
`100 100 100 2`
`10 12 14 1`
`7 8 9 10`
`8.134 9.098 7.123 5.10`
	`Case 1: 81.6496580`
	`Case 2: 7.0710678`
	`Case 3: 6.6742381`
	`Case 4: 7.4374548`

C - Problem C

Source file name: `weights.cpp`
Time limit: x seconds

The city of Bath is a noted olympic training ground—bringing local, national, and even international teams to practice. However, even the finest gymnasium falls victim to the cardinal sin... Weights put back in the wrong spots.

All of the pairs of dumbbells sit in no particular order on the two racks, possibly even with some of them split between rows. Initially each row has an equal number of dumbbells, however, this being a well-funded professional gym, there is infinite space at either end of each to hold any additional weights.

To move a dumbbell, you may either roll it to a free neighbouring space on the same row with almost no effort, or you may pick up and lift it to another free spot; this takes strength proportional to its weight. For each pair of dumbbells, both have the same unique weight.

What is the heaviest of the weights that you need to be able to lift in order to put identical weights next to each other? Note that you may end up with different numbers of weights on each row after rearranging; this is fine.

Input

The input consists of:

- one line containing the integer n ($1 \leq n \leq 10^6$), the number of pairs;
- two lines, each containing n integers $w_1, \ldots, w_n$ ($1 \leq w_i \leq 10^9$ for each i), where w_i is the mass of the weight i-th from the left along this row.

Every weight in the input appears exactly twice.

The input must be read from standard input.

Output

Output the weight of the heaviest dumbbell that must be moved, in order that all items can be paired up while lifting the smallest possible maximum weight.

The output must be written to standard output.

Sample Input	**Sample Output**
5 2 1 8 2 8 9 9 4 1 4 8 7 7 15 15 2 2 4 4 5 5 3 3 9 9 1 1	2 0