Problema A. Asteroides

Nombre código fuente:	`asteroide.c, asteroide.cpp o asteroide.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Diego Alejandro Agudelo España`

Está en curso el año 2532 y los habitantes del planeta tierra están migrando hacia al planeta *Complexus* debido principalmente a que los recursos naturales en el planeta tierra se agotan. La migración hacia *Complexus* se está llevando a cabo usando a la **UTP** (*Unidad de Transbordadores Planetarios*) y esta migración se está realizando por grupos de personas, los cuales han sido escogidos de acuerdo a su profesión. Por cuestiones del azar el primer grupo de personas en viajar fueron los científicos de la computación y el segundo grupo fueron los músicos. Cierto tiempo después del despegue del Transbordador de los músicos, estos han informado la presencia de grandes asteroides en la trayectoria hacia *Complexus* y afirman que la cantidad de asteroides es tan grande que requieren con urgencia un programa que les permita conocer cual es el asteroide más cercano a el transbordador para poder evitarlo. La mala noticia es que la comunicación con el transbordador de los científicos de la computación se ha perdido completamente, ni los músicos, ni las personas que aún se encuentra en la tierra, pueden comunicarse con los científicos de la computación. Tú eres una de las pocas personas que queda en la tierra que tiene los conocimientos y habilidades para crear este programa ¿Podrás asegurar que el Transbordador de los músicos y que todas las personas que esperan en la tierra podrán llegar sanos y salvos a *Complexus*?.

Para realizar este programa se puede asumir lo siguiente:

- El Transbordador y las Asteroides pueden ser representados en un plano.
- El Transbordador se puede representar como un punto $(X_t,\ Y_t)$.
- Los Asteroides se pueden representar como un círculo cuyo centro se encuentra en $(X_i,\ Y_i)$ y cuyo radio es R_i $(1 \leq i \leq N)$ (N representa la cantidad total de Asteroides).
- El Transbordador no estará dentro de ningún Asteroide.
- Ningún par de Asteroides se intersectarán, es decir, $R_i + R_j < \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$ $(1 \leq i,\ j \leq N)$, $i \neq j$.
- La distancia entre el Transbordador y un Asteroide se define como: la distancia mínima entre la circunferencia que representa el Asteroide y el punto que representa el Transbordador.
- No habrán empates entre las distancias entre el Transbordador y los Asteroides. Siempre habrá una diferencia de como mínimo 0.001 unidades entre las distancias mencionadas.

Entrada

La entrada contiene varios casos de prueba. La primera línea de cada caso de prueba contiene un número entero N $(1 \leq N \leq 1000)$ que representa la cantidad de Asteroides que rodean la nave. La segunda línea contiene dos números enteros X_t, Y_t $(-1000 \leq X_t,\ Y_t \leq 1000)$ que representan la posición del Transbordador. Después se presentan N líneas, cada una con tres números enteros X_i, Y_i y R_i que representan el Asteroide i-ésimo, donde:

- $-1000 \leq X_i,\ Y_i \leq 1000$
- $1 \leq R_i \leq 50$
- $1 \leq i \leq N$

La entrada termina cuando $N = 0$.

Salida

Para cada caso de prueba se debe imprimir el índice i ($1 \leq i \leq N$) que representa el Asteroide más cercano al transbordador, en una línea. La numeración de los Asteroides se hace de acuerdo al orden en que fueron presentados en la entrada.

Ejemplo

Entrada	Salida
3 0 0 0 -10 5 18 4 6 0 7 3 1 3 3 -1000 -1000 15 0	3 1

Problema B. Betty y la Potencia Modular

Nombre código fuente:	`betty.c, betty.cpp o betty.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Sebastián Gómez González`

A Betty le gusta la matemática de números enteros, y sabe que al operar a^b se pueden obtener números muy grandes cuando a y b son números grandes. A Betty solo le interesa saber cuales son los últimos 9 dígitos de la operación a^b. Por eso lo a contratado a usted, de la ***U**nidad de **T**remendas **P**otencias (**UTP**)*, para que indique los últimos 9 dígitos resultado de la potencia.

Entrada

La entrada consiste de varios casos de prueba. La primera línea contiene el número de casos de prueba T, las siguientes T líneas representan cada una un caso de prueba y tienen los 2 enteros a y b

- $1 \leq T \leq 1000$
- $1 \leq a, b \leq 10^9$

Salida

Por cada caso de prueba de la entrada, se debe imprimir las 9 cifras menos significativas del resultado de la potencia sin ceros a la izquierda en una línea independiente.

Ejemplo

Entrada	Salida
4 2 2 2 3 3 2 1001 1000	4 8 9 501000001

Problema C. Elige tu Propia Aventura

Nombre código fuente:	`aventura.c, aventura.cpp o aventura.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Leonardo Boshell`

¿Conoces los libros de la serie "*Elige tu propia aventura*"?. Son libros que cuentan historias en segunda persona, involucrando al lector como protagonista principal. Pero la característica más interesante de estos libros es que las historias pueden bifurcarse en ciertos puntos y el lector elige el camino que quiera.

Por ejemplo, imagina que la narración del libro te lleva a un bosque en donde encuentras una cueva. En este punto, el libro puede hacer una pausa y contener una nota de la siguiente forma:

> *Si decides entrar a la cueva, pasa a la página 14.*
> *Si prefieres seguir caminando por el bosque, pasa a la página 21.*

De este modo, estos libros pueden tener múltiples desenlaces, y el lector puede explorarlos a su propio ritmo si así lo desea.

Ricardo es un niño que ha recibido varios de estos libros como regalo de sus padres. A él le encantan y los devora con rapidez. Sin embargo, él se encuentra con una dificultad al momento de leerlos. él es muy curioso e impaciente, y mientras lee el libro quisiera saber cuánto falta para llegar al final de la historia, o mejor dicho, a uno de los posibles finales a los que puede llegar desde la página que está leyendo. Pero dada la estructura del libro, no le resulta fácil determinar cuántas páginas le faltan. Peor aún, si intenta ojear las páginas que no ha leído, puede llegar a arruinar varias sorpresas de la historia al ver las ilustraciones, cosa que a él no le gusta para nada.

Al enterarte de la situación, decides ayudar a Ricardo, creando un programa que reporte el número de páginas que restan para llegar al final de la historia por el camino más corto y por el camino más largo posible (en número de páginas), a partir de una página cualquiera del libro.

Para este análisis, definimos una *sección* del libro como una secuencia de una o más páginas consecutivas, que pueden leerse de forma lineal. Las siguientes reglas se aplican:

- Un libro se compone de una o más secciones.
- Una página cualquiera pertenece a una y solo una sección del libro.
- La última página de una sección contiene cero o más saltos hacia otras secciones del libro. Si no tiene saltos, es porque esa sección constituye uno de los posibles finales de la historia.
- Las secciones se enumeran desde el índice cero. Es decir, para un libro con N secciones, ellas son enumeradas: $0, 1, 2, \ldots, (N-1)$.
- El libro siempre comienza en la página 1, y la página 1 pertenece siempre a la sección cero.
- Hay una y solo una ruta posible desde la sección cero hasta cualquiera de las otras secciones del libro.

Entrada

La entrada comienza con un entero positivo T, que indica el número de casos de prueba.

Cada caso de prueba comienza con una línea en blanco, seguida de una línea con un entero N, el número de secciones de un libro. A continuación se presentan N líneas, que describen cada una de las secciones en orden (la primera de estas líneas describe la sección 0, la segunda línea la sección 1, y así sucesivamente).

Cada sección se describe con tres enteros: a, b, p, en ese orden. Los enteros a y b son los números de las páginas en donde comienza y termina una sección, respectivamente. El entero p es el número de la sección *previa*; esto es, la sección que contiene un salto a la sección siendo descrita. El valor de p para la sección cero es -1, que significa que no tiene sección previa. Se puede suponer que los casos de prueba siempre describen un libro válido de acuerdo a las reglas descritas anteriormente.

Seguidamente se presenta una línea con un entero Q, que representa el número de consultas que Ricardo va a realizar. Finalmente se presenta una línea con Q enteros, cada uno de ellos representa el número de una página válida dentro del libro.

$T \leq 100$; $1 \leq N \leq 1000$; $1 \leq a \leq b \leq 5000$; $1 \leq Q \leq 1000$

Salida

Por cada caso de prueba, imprime en una línea **Caso *i*:**, en donde i representa el número del caso. A continuación deben reportarse los resultados de las Q consultas, cada una en una línea diferente y en el mismo orden de la entrada.

Cada una de estas líneas debe contener dos números: el número mínimo y el número máximo de páginas que restan por leer hasta un posible final de la historia, a partir de la página indicada como entrada. La página de la consulta debe incluirse (ser contada) en los resultados. Para más detalles, consulte los ejemplos a continuación.

Ejemplo

Entrada	Salida
2 3 1 13 -1 14 20 0 21 24 0 4 1 13 14 22 4 1 9 -1 20 29 2 10 19 0 30 33 2 2 19 3	Caso 1: 17 20 5 8 7 7 3 3 Caso 2: 5 11 21 27

Problema D. El Cajón de las Medias (I)

Nombre código fuente:	`mediasuno.c, mediasuno.cpp o mediasuno.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Leonardo Boshell`

Carolina es una estudiante que debe levantarse muy temprano algunos días para asistir a sus clases. Ella es un poco desorganizada, de modo que vestirse por las mañanas suele resultarle un poco complicado.

Por ejemplo, ella tiene un cajón especial para sus medias, pero no las dobla ni las coloca ordenadamente en el mismo. Simplemente arroja sus medias limpias al cajón, y cuando tiene que vestirse saca medias al azar hasta que obtiene un par apropiado. Por lo general ella se viste con dos medias del mismo color, aunque a veces se le antoja cambiar su pinta, y prefiere vestirse con medias de colores diferentes.

Entre su colección, Carolina tiene medias de K colores diferentes. La cantidad exacta de medias en el cajón de un color en particular puede variar, y se denotará c_i, siendo i el numeral que identifica a un color: $1 \leq i \leq K$.

Conociendo los valores de K y $c_1,\ c_2,\ \ldots,\ c_K$, y suponiendo que ella saca las medias del cajón completamente al azar, ¿cuál es el mínimo número de medias que tiene que sacar para garantizar que tenga al menos 2 del *mismo* color?, ¿cuál es el mínimo número para garantizar que tenga al menos 2 de *diferente* color?.

Entrada

La entrada comienza con un entero positivo T, que indica el número de casos de prueba.

La primera línea de cada caso contiene el entero K. La segunda línea contiene K enteros, que forman la secuencia $c_1,\ c_2,\ \ldots,\ c_K$.

$T \leq 1000$; $2 \leq K \leq 15$; $2 \leq c_i \leq 30$

Salida

Por cada caso de prueba, imprime **Caso *i*:**, en donde i representa el número del caso, seguido de dos números enteros: el mínimo número de medias que debe sacar para garantizar que tiene 2 del mismo color, y el mínimo número de medias que debe sacar para garantizar que tiene 2 de colores diferentes, en ese orden.

Ejemplo

Entrada	Salida
1 2 2 2	Caso 1: 3 3

Problema E. Exponente

Nombre código fuente:	`exponente.c, exponente.cpp o exponente.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Diego Alejandro Martínez Duque`

Dado un conjunto de números dados por una base y un exponente, determinar cual es el mayor de ellos.

Entrada

La entrada consiste de multiples casos de prueba, para cada caso de prueba, en la primera línea un entero n ($1 \leq n \leq 6000$), que indica el número de pares, seguido por n líneas, cada una con 2 enteros b y e ($1 \leq b \leq 9$, $1 \leq e \leq 1000$), base y exponente respectivamente, la entrada finaliza con n = 0, el cual no debe ser procesado.

Salida

En una línea independiente mostrar el máximo valor de cada caso de prueba.

Ejemplo

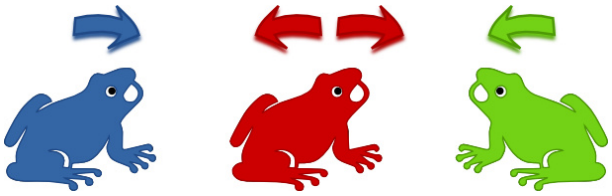
Entrada	Salida
3 7 28 5 32 9 22 0	459986536544739960976801

Problema F. Ranas Saltarinas

Nombre código fuente:	`ranas.c, ranas.cpp o ranas.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Leonardo Boshell`

Raúl y Jaime son dos hermanos jóvenes y astutos que pasan las tardes jugando. Cierto día, buscando entre sus juguetes viejos, encuentran varias ranas de plástico de colores. Al verlas, a Raúl se le ocurre una idea para un nuevo juego.

En primer lugar, Raúl dibuja sobre un pliego de papel una hilera de 8 cuadros, que conforma el "tablero" del juego. Luego coloca 6 ranas cualquiera sobre el tablero (cada rana cubre un cuadro), en posiciones al azar (ésta es la formación inicial). Hay tres tipos de ranas diferentes: azules, rojas y verdes. El objetivo del juego es mover las ranas hasta que produzcan una formación final arbitraria, elegida con anterioridad.



Tipos de rana y sus movimientos

Como reglas del juego, las ranas azules sólo pueden moverse hacia la derecha, las verdes sólo pueden moverse hacia la izquierda, y las rojas pueden moverse en cualquiera de estas dos direcciones. Estos movimientos pueden tener un rango de uno o dos cuadros. En el primer caso, la rana salta a un cuadro adyacente si éste se encuentra vacío. En el segundo caso, la rana puede saltar dos cuadros únicamente si el salto es sobre una rana ubicada en un cuadro adyacente, y hay un cuadro vacío inmediatamente después de la rana saltada.

Raúl le explica el juego a Jaime, y le propone que, después de ponerse de acuerdo sobre las formaciones de inicio y fin, los dos intenten resolverlo en el menor número de movimientos posible. Tu misión es escribir un programa que resuelva este problema.

Entrada

La entrada comienza con un entero positivo T ($T \leq 500$), que indica el número de casos de prueba.

Cada caso de prueba es descrito en tres líneas. La primera es una línea en blanco, mientras que las dos siguientes líneas describen las formaciones inicial y final, en ese orden. Cada formación es descrita por 8 caracteres que representan los cuadros del tablero. Las letras A, R y V representan ranas de color azul, rojo y verde, respectivamente, mientras que un punto representa un cuadro vacío.

Cada formación contiene exactamente 6 letras y 2 puntos. Para cada caso de prueba, la formación final siempre contiene el mismo número de ranas de cada color que la formación inicial.

Salida

Por cada caso de prueba, imprime **Caso *i*:**, en donde i representa el número del caso, seguido de un espacio y la respuesta. Si el juego tiene solución, imprime el mínimo número de movimientos requerido para resolver el juego de las ranas saltarinas. Si el juego no tiene solución, imprime **imposible**.

Ejemplo

Entrada	Salida
3 AV.A.RVV VRA..AVV AAA..VVV VVV..AAA .AAARRR. ARR..RAA	Caso 1: 6 Caso 2: 18 Caso 3: imposible

Problema G. ¿Cuántos Números Primos?

Nombre código fuente:	`primos.c, primos.cpp o primos.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Santiago Gutiérrez Alzate`

Recordar la definición de *número primo*: un número entero positivo p más grande que uno es llamado número primo si y sólo si tiene como divisores únicamente a los números enteros positivos 1 y p.

Dados dos números enteros positivos a y b, calcular cuántos números primos existen en el intervalo cerrado $[a,\ b]$.

Entrada

La entrada puede contener varios casos de prueba. Cada caso de prueba se presenta en una línea independiente y contiene dos números enteros positivos a y b ($2 \leq a \leq b \leq 10^6$). La entrada finaliza con un caso de prueba en el que tanto a y b tiene el valor de 0, caso que no debe ser procesado.

Salida

Por cada caso de prueba, imprimir una única línea con la cantidad de números primos p_i que cumplan que $a \leq p_i \leq b$.

Ejemplo

Entrada	Salida
4 4	0
5 5	1
4 6	1
2 10	4
2 23	9
10 20	4
0 0	

Problema H. La Sucesión de Humbertomb Moralomb

Nombre código fuente:	`humberto.c, humberto.cpp o humberto.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Hugo Humberto Morales Peña`

La sucesión de Humbertomb Moralomb (HM) se apoya en la sucesión de los números enteros impares positivos $I(n)$:

n	1	2	3	4	5	6	7	8	9	$\cdots$
$I(n)$	1	3	5	7	9	11	13	15	17	$\cdots$

En la *sucesión de Humbertomb Moralomb* el valor n está $I(n)$ veces, de esta forma se tienen los primeros 16 términos en dicha sucesión:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	$\cdots$
$I(n)$	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	$\cdots$
$HM(n)$	1	2	2	2	3	3	3	3	3	4	4	4	4	4	4	4	$\cdots$

En este problema se debe escribir un programa que calcule el valor que está en la posición n en la sucesión de *Humbertomb Moralomb* (es decir, que calcule el $HM(n)$).

Entrada

La entrada puede contener varios casos de prueba. Cada caso de prueba se presenta en una línea independiente y contiene un entero n ($1 \leq n \leq 2.000.000.000$). La entrada finaliza con un caso de prueba en el que n tiene el valor de 0, caso que no debe ser procesado.

Salida

Por cada caso de prueba de la entrada, se debe imprimir el valor de $HM(n)$ en una línea independiente.

Ejemplo

Entrada	Salida
100 9999 123456 1000000000 0	10 100 352 31623

Problema I. Hecho en Colombia

Nombre código fuente:	`colombia.c, colombia.cpp o colombia.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Sebastián Gómez González`

Los libros estadísticos de todas partes del mundo especifican como calcular la media y la mediana de cualquier conjunto de datos. La media o promedio de un conjunto de datos $X = \{X_0,\ X_1,\ \ldots,\ X_{n-1}\}$ se define como:

$$\mu = \sum_{i=0}^{n-1} \frac{X_i}{n}$$

La mediana del mismo conjunto de datos, es el dato en la posición $\lfloor \frac{n}{2} \rfloor$ del arreglo X ordenado de forma ascendente. Veamos el siguiente ejemplo:

I	Datos	Ordenado	Mediana
1	$\{3\}$	$\{3\}$	3
2	$\{3,\ 2\}$	$\{2,\ 3\}$	2
3	$\{3,\ 2,\ 1\}$	$\{1,\ 2,\ 3\}$	2
4	$\{3,\ 2,\ 1,\ 8\}$	$\{1,\ 2,\ 3,\ 8\}$	2
5	$\{3,\ 2,\ 1,\ 8,\ 7\}$	$\{1,\ 2,\ 3,\ 7,\ 8\}$	3

En el ejemplo anterior, la mediana con $I = 5$ es el elemento en la posición $\lfloor \frac{5}{2} \rfloor = 2$ del arreglo ordenado, es decir, el número 3. En Colombia, para diferenciarnos del resto de países, se toma la decisión de incluir en los libros estadísticos como calcular la media de las medianas. Si $M(X)$ calcula la mediana de los datos del conjunto X, la media de las medianas μ_m de un conjunto de datos $X = \{X_0,\ X_1,\ \ldots,\ X_{n-1}\}$ se calcula así:

$$\mu_m = \frac{1}{n} \sum_{i=0}^{n-1} M(\{X_0,\ X_1,\ \ldots,\ X_i\})$$

Para explicar mejor como se hace el cálculo, supongamos que el conjunto de datos X es $\{3,\ 2,\ 1,\ 8,\ 7\}$. Como la entrada tiene 5 valores, se deben calcular 5 medianas. La tabla anterior muestra el ejemplo del cálculo de las 5 medianas sobre las que se debe calcular el promedio, que representan las medianas después de insertar cada uno de los 5 valores. Para calcular la media de las medianas, simplemente se toma el promedio de las medianas calculadas:

$$\mu_m = \frac{3 + 2 + 2 + 2 + 3}{5} = 2.4$$

En **INSILICO** (**IN**vestigaciones **S**obre **I**nformática y **LI**bros de **CO**lombia), lo han contratado a usted para que implemente un programa que pueda calcular la media de las medianas de un conjunto de datos.

Entrada

La entrada consiste de varios casos de prueba. La primera línea de cada caso de prueba contiene un entero N que indica la cantidad de elementos del conjunto de entrada X, luego vienen N enteros que indican cada uno de los elementos desde X_0 hasta X_{n-1}. El último caso de prueba va seguido por un caso con $N = 0$, el cual no debe ser procesado.

- $1 \leq N \leq 10^6$
- $0 \leq X_i \leq 10^9 \quad \forall i \in [0, N-1]$

Salida

Por cada caso de prueba de la entrada, se debe imprimir en una sola línea el valor de la media de medianas redondeada a 2 cifras decimales.

Ejemplo

Entrada	Salida
5 3 2 1 8 7 0	2.40

Problema J. Los Números Triangulares

Nombre código fuente:	`triangular.c, triangular.cpp o triangular.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Hugo Humberto Morales Peña`

Los *Números Triangulares* son todos aquellos números enteros positivos que representan una cantidad de asteriscos que pueden conformar un triángulo compacto con la misma cantidad de asteriscos en cada uno de sus tres lados.

Los primeros cinco números triangulares son:

1	3	6	10	15
*	* * *	* * * * * *	* * * * * * * * * *	* * * * * * * * * * * * * * *

En este problema se debe escribir un programa que determine si un número n es triangular o no.

Entrada

La entrada puede contener varios casos de prueba. Cada caso de prueba se presenta en una línea independiente y contiene un entero n ($1 \leq n \leq 16 \times 10^{18}$). La entrada finaliza con un caso de prueba en el que n tiene el valor de 0, caso que no debe ser procesado.

Salida

Por cada caso de prueba de la entrada, se debe imprimir un **SI** o un **NO** dependiendo si el número de la entrada es o no triangular. Cada caso valido de entrada debe generar una línea de salida.

Ejemplo

Entrada	Salida
1 15 16 101 0	SI SI NO NO

Problema K. El Cajón de las Medias (II)

Nombre código fuente:	`mediasdos.c, mediasdos.cpp o mediasdos.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Leonardo Boshell`

Es una linda mañana, y Carolina se dirige a la Universidad, en donde le espera un examen de *Estadística y Probabilidad*. Ella ha estado estudiando para este examen durante varios días, y en su camino va pensando sobre la *esperanza matemática*, cuando de repente le viene a la cabeza la imagen de su rutina diaria al momento de elegir las medias que se va a poner.

Recordemos que Carolina tiene medias de K colores diferentes, y la cantidad de medias de cada color se denota $c_1,\ c_2,\ \ldots,\ c_K$. Carolina decide entonces realizar un ejercicio mental como práctica para su examen, y se dispone a calcular el número *esperado* de medias que tiene que sacar de su cajón hasta tener en sus manos 2 del mismo color. ¿Puedes crear un programa que le ayude a responder esta pregunta?

Se debe suponer que Carolina saca sus medias completamente al azar del cajón —la probabilidad de elegir una media del cajón en cualquier momento es la misma para todas las medias.

Entrada

La entrada comienza con un entero positivo T, que indica el número de casos de prueba.

La primera línea de cada caso contiene el entero K. La segunda línea contiene K enteros, que forman la secuencia $c_1,\ c_2,\ \ldots,\ c_K$.

$T \leq 100\ \ ;\ \ 1 \leq K \leq 15\ \ ;\ \ 2 \leq c_i \leq 30$

Salida

Por cada caso de prueba, imprime **Caso *i*:**, en donde i representa el número del caso, seguido del número esperado de medias que Carolina debe sacar de su cajón hasta tener dos medias del mismo color.

Imprime el resultado como un número real con exactamente 3 dígitos después del punto decimal.

Ejemplo

Entrada	Salida
3 2 2 2 3 10 20 30 5 2 3 5 7 11	Caso 1: 2.667 Caso 2: 2.797 Caso 3: 3.343

Problema L. Mediana Simple

Nombre código fuente:	`mediana.c, mediana.cpp o mediana.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Sebastián Gómez González`

La *Mediana en Estadística* es un indicador que se obtiene de la siguiente forma. Se toman los N datos a los que se les desea sacar la mediana en orden ascendente. Si el número de datos N es impar, se toma el dato de la mitad. Por ejemplo la secuencia 4, 1, 3, ordenada queda 1, 3, 4 y su mediana es el valor de la mitad 3. Cuando la cantidad de datos es par, no hay un único valor en la mitad. Por ejemplo la secuencia 3, 2, 6, 1 ordenada queda 1, 2, 3, 6 y los valores de la mitad serán 2 y 3, la mediana se toma como el promedio de los dos valores de la mitad, entonces queda $\frac{2+3}{2} = 2.5$. En este problema se solicita que calcule la mediana de un conjunto de datos.

Entrada

La entrada consiste de varios casos de prueba. La primera línea de cada caso de prueba es el número N que indica a cuantos números hay que calcular la mediana. La siguiente línea contiene los N enteros x_i separados por un espacio indicando a los que se debe calcular la mediana. El fin de la entrada viene por una línea con $N = 0$ que no debe ser procesada.

- $1 \leq N \leq 10^5$
- $1 \leq x_i \leq 10^9$

Salida

Por cada caso de prueba de la entrada, se debe imprimir el valor de la mediana redondeada a una cifra decimal en una línea independiente.

Ejemplo

Entrada	Salida
1 5 2 5 4 3 5 5 4 0	5.0 4.5 5.0

Problema M. El Cultivo de Don UTP

Nombre código fuente:	`donutp.c, donutp.cpp o donutp.java`
Entrada:	`Estándar`
Salida:	`Estándar`
Autor(es):	`Diego Alejandro Agudelo España`

Don *Uriel Toussaint Pratt* es un conocido granjero de la ciudad de *Sumtab* (es conocido normalmente entre los habitantes de *Sumtab* como Don **UTP**). Don UTP está incursionando en el negocio del cultivo de alimentos ricos en vitaminas para mejorar las habilidades algorítmicas de las personas que los consumen. Don UTP, en la búsqueda constante de recetas que le ayuden a mejorar sus productos, ha consultado un grupo de laboratorios que le han proporcionado una serie de fórmulas que especifican las vitaminas que debe tener un alimento para mejorar el pensamiento algorítmico de las personas que lo consumen. Cada fórmula F es una secuencia de letras minúsculas $c_1,\ c_2,\ \ldots,\ c_n$ donde cada letra c_i representa una vitamina que debe estar presente en el alimento de acuerdo a la fórmula F (todas las vitaminas en una fórmula F son distintas), es importante mencionar que los laboratorios le garantizaron a Don UTP que para que cierta fórmula funcione en un alimento, este debe contener absolutamente todas las vitaminas especificadas en la fórmula y solamente esas vitaminas, de lo contrario, el alimento podría causar efectos secundarios en sus consumidores.

Don UTP posee un cultivo diverso de vitaminas algorítmicas. Este cultivo es rectangular y se puede representar como una cuadrícula de N filas por M columnas, donde cada celda contiene una vitamina particular (denotada por una letra minúscula). Don UTP ha desarrollado una nueva estrategia de cosecha muy eficiente que le permite cosechar cualquier porción rectangular de su cultivo (incluso el cultivo completo).

Don UTP tiene el siguiente interrogante ¿Cuál es el área de la porción rectangular más grande de su cultivo que contiene todas las vitaminas especificadas por una fórmula dada F y ninguna otra vitamina?. Don UTP necesita de tu ayuda, él sabe que has sido un consumidor regular de sus alimentos.

Entrada

La entrada contiene varios casos de prueba. La primera línea de cada caso de prueba contiene un par de números enteros N y M ($1 \leq N,\ M \leq 200$) que representan respectivamente el número de filas y de columnas del cultivo de Don UTP. La segunda línea contiene una cadena de letras minúsculas F ($|F| \leq 26$) que representa la fórmula dada por los laboratorios como se explicó en el enunciado. después se presentan N líneas, cada una con M letras minúsculas que representan el cultivo de Don UTP (ver los ejemplos para ver el formato exacto). La entrada termina cuando $N = M = 0$.

Salida

Para cada caso de prueba se debe imprimir el área máxima que busca Don UTP en una línea.

Ejemplo

Entrada	Salida
4 5 utp abutp unutp ttutp pputp 3 3 a aaa aaa aaa 0 0	12 9