Universidad Nacional de San Antonio Abad del Cusco
Facultad de Ingenieria Electrica, Electronica, Informatica y Mecanica
Escuela Profesional de Ingenieria Informatica y de Sistemas

Concurso de Programación
X CusContest

PROBLEM SET

Este conjunto de problemas contiene 8 problemas,
páginas numeradas de 1 a 10 .

Organizado por:
ACM Chapter Cusco
ACM Chapter Cusco
Escuela Profesional de Ingeniería Informática y de Sistemas
Departamento Académico de Informática - UNSAAC
Centro Federado de Ingeniería Informática y de Sistemas
Curso de Actividades 2016-I

Julio 20 , 2016
CUSCO - PERÚ

INFORMACIÓN GENERAL

Salvo se indique lo contrario, lo siguiente vale para todos los problemas.

Entrada

- La entrada se debe leer de la entrada estándar (*standard input*).
- Cuando una línea de datos contiene varios valores, estos se separan utilizando exáctamente un espacio entre ellos. Ningún otro espacio aparece en la entrada.
- No hay letras con tildes, acentos, diéresis, ni otros signos ortográficos (ñ, Ã, ç, ú, æ, etcétera).

Salida

- La salida se debe escribir en la salida estandar (*standard output*).
- El resultado del caso de prueba debe de aparecer en la salida utilizando una cantidad de líneas que depende del problema. No debe de haber otros datos en la salida.
- Cuando una línea de resultados contiene varios valores, estos se deben de separar utilizando exáctamente un espacio entre ellos. Ningún otro espacio debe de aparecer en la salida.
- No debe de haber letras con tildes, acentos, diéresis, ni otros signos ortográficos (ñ, Ã, ç, ú, æ, etcétera).

Tiempo Límite

- El tiempo límite informado corresponde al tiempo total permitido para la ejecución de una solución sobre todo el conjunot de casos de prueba de un determinado problema.

Equipo de desarrollo:
Alexander Pinares, Federal University of Rio Grande do Sul
Berthin Torres, University of Campinas
Grover Castro, University of São Paulo
Javier Vargas, University of Campinas

Problem A. Horarios

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

En la carrera profesional de Ingeniería Informática y de Sistemas muchos alumnos llevan cursos en diferentes horarios. Algunos cursos pueden ser en la mañana, tarde o noche y no siempre se tiene un horario consecutivo.

Warnoldo, es un estudiante de la carrera que estaba preocupado en determinar cuanto es el máximo intérvalo libre y el máximo intervalo consecutivo de cursos que tiene. No es difícil averiguar para una sóla persona dicha información, pero muchos estudiantes al enterarse de la idea de Warnoldo también sienten curiosidad para determinar dichos valores. Por tanto, ahora se necesita diseñar un programa que realice la tarea mencionada.

Dados N cursos y el intérvalo de tiempo (en minutos) de cada curso, tu tarea es determinar el máximo intérvalo libre entre cursos(t_{libre}) y el máximo intérvalo donde estarás ocupado en los cursos($t_{ocupado}$). Por conveniencia, todos los cursos comienzan a las 7am, los minutos se contarán a partir de las 7am y por tanto, la información de cada curso tendrá un `tiempo_inicio` (t_o) y `tiempo_fin` (t_f) ambos en minutos. Si un curso comienza a las 7am y termina a las 9am, t_o será igual a 0 y t_f igual a 120.

Existe la posibilidad de que los horarios para algunos cursos se sobrepongan.

Input

La entrada del problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 10$) indicando el número de casos de prueba. Para cada caso, se sigue el siguiente formato de entrada:

- La primera línea contiene contiene un número entero N indicando la cantidad de cursos ($0 < N \leq 5 \times 10^3$).
- Las siguientes N líneas contienen dos enteros t_o y t_f indicando el tiempo de inicio y fin de cada curso ($0 \leq t_o < t_f$).

Output

Para cada caso de prueba, el programa deberá imprimir una línea mostrando los valores de t_{libre} y $t_{ocupado}$ por un espacio en blanco.

Example

Standard input	Standard output
2 2 0 120 240 360 3 30 100 70 120 150 210	120 120 30 90

Problem B. Mostrando grafos

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

En la clase de algoritmos y estructuras de datos, el profesor estaba explicando el tema de árboles y Gerardo (uno de sus estudiantes) lo interrumpió para preguntar que tipos de recorridos en árboles existían. Como faltaban 5 minutos para el final, el profesor decidió dejarles de tarea hacer un algoritmo que recorra un árbol por niveles.

Dado que ya no tenía mucho tiempo para explicar todos los detalles dejó en la pizarra la siguiente explicación sobre un árbol:

Un árbol $T = (V, E)$ está conformado por un conjunto de nodos $V = \{0, 1, \ldots, N-1\}$, y un conjunto de aristas $E = \{(p, q)/p, q \in V\}$. El árbol tiene un nodo raíz $r \in V$ y tiene exáctamente $N - 1$ aristas. Cada elemento $(p, q) \in E$, indicará que existe arista desde el nodo p hacia el nodo q.

Además, considerando el siguiente conjunto de nodos $V = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$ y el conjunto de aristas $E = \{(4, 3), (4, 6), (4, 0), (3, 1), (3, 7), (3, 9), (6, 2), (6, 8), (0, 11), (1, 5), (1, 10)\}$, se puede determinar que la raíz del árbol es el nodo 4 y el árbol tiene 4 niveles. Sin embargo, es posible formar diferentes árboles dependiendo del órden en el que se procesen los nodos. Por ejemplo, con esas aristas pueden formarse los dos árboles siguientes:

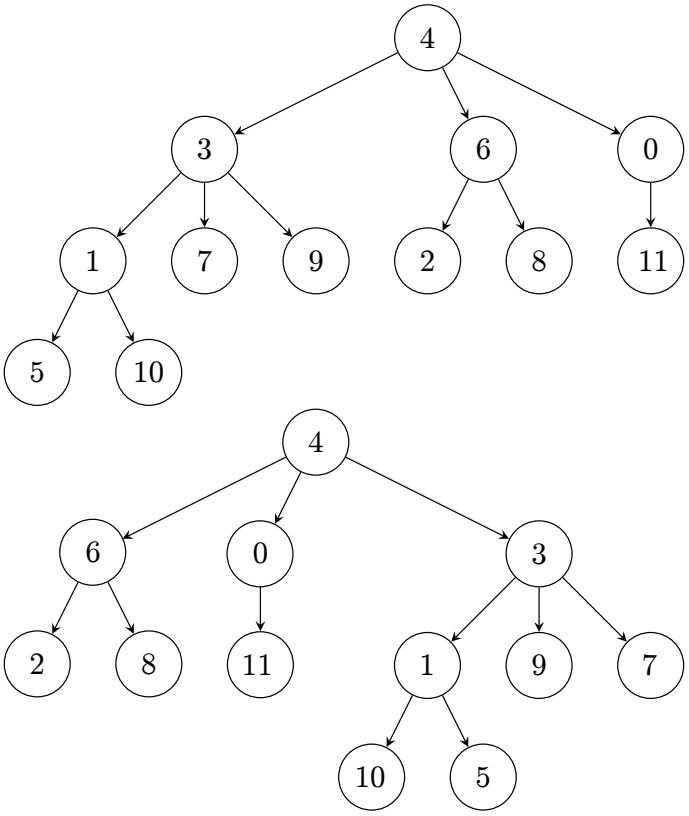

Figura. 1: Dos árboles que pueden armados considerando la configuración explicada en el párrafo anterior.

Tu deber es armar un árbol y recorrerlo por niveles. Es decir, considerando el árbol que aparece en la parte superior, primero se muestra la raiz (nodo 4), luego sus hijos (nodos 3, 6, y 0), después los hijos de estos nodos que se encuentran en el siguiente nivel (nodos 1, 7, 9, 2, 8, y 11), y así sucesivamente. Por consiguiente, la respuesta que se busca sería la siguiente:

```
4
3 6 0
1 7 9 2 8 11
5 10
```

Pero dependiendo de como se arme el árbol se pueden tener varias respuestas válidas. Para nuestro ejemplo, otra respuesta válida considerando el árbol inferior es:

```
4
6 0 3
2 8 11 1 9 7
10 5
```

Input

La entrada del problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 100$) indicando el número de casos de prueba. Cada caso de prueba sigue el siguiente formato de entrada:

- La primera línea contiene un entero N indicando la cantidad de nodos del árbol ($1 \leq N \leq 10^3$).
- La segunda línea contiene $N-1$ pares de enteros p, q representando las aristas del árbol ($0 \leq p, q < N, p \neq q$). Por claridad, un par de enteros p, q expresa una arista del nodo p hacia el nodo q.

Output

Para cada caso de prueba, el programa deberá imprimir "`Caso #i:`" (sin comillas) y a partir de la siguiente linea el árbol por niveles. Como pueden haber varios recorridos por niveles válidos para un mismo conjunto de aristas, imprimir cualquier recorrido válido.

Example

Standard input	Standard output
3 2 0 1 5 0 1 1 2 2 3 3 4 12 4 3 3 1 6 2 0 11 3 9 4 0 1 5 6 8 1 10 3 7 4 6	Caso #1: 0 1 Caso #2: 0 1 2 3 4 Caso #3: 4 3 6 0 1 7 9 2 8 11 5 10

Problem C. Cuadradolandia

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

En el país de Cuadradolandia, las construciones se realizan usando figuras geométricas muy especiales. Los habitantes de esta ciudad se divierten contando el número de cuadrados, cubos, hipercupos u otras figuras geométricas de tamaño regular.

Cierto día Warrencio, el más hábil de la ciudad se pregunta sí hay una forma de calcular el número de cuadrados para una pared de $N \times N$ o el número de cubos en un edificio de $N \times N \times N$ o el número de hipercubos en un edificio de $N \times N \times N \times N$ o inclusive para edificaciones en espacios D-dimensionales. Tu tarea es ayudar a Warrencio en su noble tarea.

Input

La entrada del problema contiene varios casos de prueba. La primera línea es un entero T indicando el número de casos de prueba. Para cada caso, se sigue el siguiente formato de entrada:

- Una línea que contiene 2 enteros M y D siendo M el número de figuras geométricas regulares de dimensión $\underbrace{1 \times 1 \times \cdots \times 1}_{D \text{ veces}}$ en un espacio D-dimensional ($1 \leq M \leq 16807$ y $1 \leq D \leq 5$).

Output

Para cada caso de prueba, imprimir en una línea el máximo número de figuras geométricas regulares que se pueden formar.

Example

Standard input	Standard output
3 4 2 8 3 81 4	5 9 98

Observación

Considerando un muro en 2 dimensiones de tamaño 3×3, el número de figuras geométricas regulares de dimensión 1×1 es igual a 9.

Problem D. Invertidor de Cadenas

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Reciéntemente, un grupo estudiantil decidió digitalizar todos los documentos de texto en la universidad. Xavier, estudiante de Informática quiso agilizar todo el procedimiento mediante algoritmos de reconocimiento de texto en imágenes. Sin embargo, su algoritmo tenía un error que invertía las oraciones. Por ejemplo, uno de los documentos que necesitan ser digitalizado contenía el siguiente texto:

```
La derrota no es el peor de los fracasos.  No
intentarlo es el verdadero fracaso. La  unica
decision  valida  es  no  rendirse  y  seguir
continuando la pelea.
```

Pero el algoritmo de Xavier retornaba:

```
Fracasos  los  de  peor  el es no derrota la.
Fracaso  verdadero  el  es  intentarlo  no.
Pelea la continuando seguir y rendirse no  es
valida decision unica la.
```

Ayuda a Xavier diseñando un algoritmo que reciba como entrada una oración y genere la oración invertida correspondientemente.

Por simplicidad, dentro de una oración sólo el primer caracter de la primera palabra estará en mayúscula, y esto debe de cumplirse siempre. No habrán signos de puntuación a excepción del punto final que indica el fin de una oración. Considerar que entre dos palabras sólo habrá un espacio en blanco de separación.

Input

La entrada del problema contiene varios casos de prueba. La primera línea es un entero T indicando el número de casos de prueba. Para cada caso, se sigue el siguiente formato de entrada:

- Una línea contieniendo la oración cuyo órden de palabras debe de ser invertido. La longitud total de la oración no será mayor a 10^3 caracteres contando los espacios en blanco y el punto final.

Output

Para cada caso de prueba, el programa deberá imprimir en una línea la oración en el órden correcto.

Example

Standard input	Standard output
3 Facil ser de antes dificil es todo. Quieres concurso el ganar si. Debes pregunta esta resolver.	Todo es dificil antes de ser facil. Si ganar el concurso quieres. Resolver esta pregunta debes.

Problem E. Warrencio y el Dota

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Cierto día Warrencio y sus N amigos fueron al BZ a jugar Dota. Cada uno tiene preferencia de usar algunas máquinas y no puede usar otras aunque estén disponibles — al final un jugador sólo puede usar a lo más una máquina pero puede preferir varias. Al llegar a recinto de la sabiduría y las buenas palabras notaron que sólo disponen de M máquinas. Por lo tanto, ayuda a Warrencio a calcular cuantos de sus amigos jugarán con él. Warren siempre va a usar una máquina ya que el vicio le hizo tener un trato especial con el dueño del internet.

Input

La entrada del problema contiene varios casos de prueba. La primera línea es un entero T indicando el número de casos de prueba. Para cada caso, se sigue el siguiente formato de entrada:

- La primera línea contiene 3 enteros N, M y P. N es el número de amigos de Warren, M es el número de máquinas y P es el número de preferencias de todos los jugadores ($0 \leq N \leq 99$, $0 \leq M \leq 100$ y $0 \leq P \leq N \times M$).
- Las siguientes P líneas contienen 2 enteros u y v siendo u el índice de los jugadores y v es el índice de las máquinas ($0 \leq u \leq N$ y $0 \leq v < M$). Warrencio tiene el índice 0

Output

Para cada caso se debe imprimir el número máximo de amigos de Warrencio que tienen una máquina para jugar.

Example

Standard input	Standard output
1 3 4 5 0 2 1 0 2 3 3 1 3 2	3

Problem F. El Baile de Invierno

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Hogwarts fue escogida como sede de la mayor competencia entre magos del mundo, el Torneo de los Tres Magos, que envolverá las tres escuelas de magia más famosas de Europa: Hogwarts, Beauxbatons y Durmstrang. Este torneo funciona de la siguiente forma. Inicialmente, un representante (llamado campeón) de cada escuela es seleccionado a través del Cáliz de Fuego. Después, los campeones realizan, a lo largo del año, tres tareas diferentes. Se consagra de vencedor del Torneo de los Tres Magos el campeón que acumula más puntos en las tres tareas.

Parte tradicional del Torneo de los Tres Magos, el Baile de Invierno se llevará a cabo en el Gran Salón, en el día de Navidad. La profesora Minerva McGonagall fue designada como responsable por las aulas de baile de los alumnos de Hogwarts. A fin de prepararlos para el Baile de Invierno, después del término de aulas muchas parejas fueron formadas voluntariamente. Sin embargo algunos alumnos permanecieron sin pareja. Por esta razón, la profesora Minerva decidió formar parejas por cuenta propia.

Ella cree que la mejor forma de emparejar es unir un hombre y una mujer que tengan, como máximo, una pequeña diferencia entre sus habilidades de baile. Dado que la profesora Minerva conoce la habilidad de baile (representado por número entero entre 1 y 100, tal que el número 1 se refiere a un pésimo bailarín y el número 100 se refiere a un bailarín perfecto) de cada uno de sus alumnos, ella intentará formar el mayor número de parejas tales que, para cada pareja, la diferencia de habilidades de baile entre el hombre y la mujer sea menor o igual a un número entero D. Las parejas formadas deste modo son llamadas "parejas equilibradas".

Tu tarea es escribir un programa que, dadas las habilidades de baile de los alumnos sin pareja y la diferencia máxima de habilidades permitida entre los integrantes de una pareja, determinar el mayor número de parejas equilibradas que pueden ser formadas.

Input

La entrada del problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 100$) indicando el número de casos de prueba. Para cada caso, se sigue el siguiente formato de entrada:

- La primera contiene tres números enteros M, N, y D tales que M representa el número de hombres, N el número de mujeres, y D la diferencia máxima de habilidades permitida entre cada integrante de una pareja ($1 \leq M, N \leq 10^5, 0 \leq D \leq 99$).
- La segunda línea contiene M números enteros $X_1, X_2, \ldots, X_M$ que representan las habilidades de los hombres ($1 \leq X_i \leq 100$).
- La tercera línea contiene N números enteros $Y_1, Y_2, \ldots, Y_N$ que representan las habilidades de las mujeres ($1 \leq Y_i \leq 100$).

Output

Para cada caso de prueba, el programa deberá imprimir una línea con el formato "`# maximo de parejas equilibradas =` R", donde R indica el mayor número de parejas equilibradas que pueden ser formadas.

Example

Standard input	Standard output
2 8 7 4 5 15 25 35 45 55 65 75 10 20 30 40 50 60 70 12 14 0 46 41 51 45 36 48 49 51 51 39 59 57 52 39 55 50 54 30 51 33 30 52 55 31 30 43	# maximo de parejas equilibradas = 0 # maximo de parejas equilibradas = 2

Problem G. El viaje de Adam

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Adam es una persona a quien le gusta conocer nuevos lugares, actualmente él se encuentra en la ciudad 1 y quiere llegar a la ciudad de $N + 1$ pasando por N sitios durante su recorrido. Adam sabe que para ir de la ciudad i a la ciudad $i + 1$ existen a_i formas de llegar. Adam desea saber el número de caminos distintos de llegar a la ciudad $N + 1$ desde la ciudad 1. Sin embargo Adam solo desea saber el primer y último dígito del número de caminos posibles, ya que el número de caminos puede ser un número muy grande.

Input

La entrada del problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 10^3$) indicando el número de casos de prueba. Para cada caso, se sigue el siguiente formato de entrada:

- La primera línea contiene un entero N indicando el número de ciudades por las cuales Adam debe de pasar ($1 \leq N \leq 10^3$).
- La segunda línea contiene N enteros distintos $a_1, a_2, \ldots, a_N$ representando el número de caminos posibles para ir de la ciudad i a la ciudad $i + 1$ ($1 \leq a_i \leq 10^9$).

Output

Para cada caso de prueba, el programa deberá imprimir el primer y último dígito de número de caminos posibles para ir de la ciudad 1 a la ciudad N.

Example

Standard input	Standard output
3 1 10 3 3 5 7 2 123 456	1 0 1 5 5 8

Problem H. Un viaje de ensueño

Input file:	Standard input
Output file:	Standard output
Time limit:	1 second
Memory limit:	64 megabytes

Warren y sus amigos piensan ir de paseo hacia Arequipa, para celebrar sus buenas notas en todos los cursos. Ellos contrataron un omnibus interprovincial para realizar el viaje, hicieron el contrato para viajar a las $5:00$ am y todos debian estar puntuales.

Llegado el día del viaje Warren se quedó dormido así que se apresuro en alistarse, y como es una persona ahorradora decidió irse en combi pues confiaba que llegaría a tiempo. Cuando llegó terminal terrestre, el omnibus al que debía embarcar ya había salido y se encontraba a una distancia d del terminal (considerar que todo el trayecto es una línea recta). Sin embargo, aún tiene la opción de realizar el viaje con sus amigos tomando un taxi e intentar dar alcance al omnibus. El taxi va a una velocidad constante de v_t y Waren quiere saber si es posible alcanzar al omnibus que tiene una velocidad constante de v_o, y si fuera así en cuanto tiempo podría alcanzar al bus.

Ayuda a Warren con su dilema.

Input

La entrada del problema contiene varios casos de prueba. La primera línea es un entero T ($1 \leq T \leq 10^4$) indicando el número de casos de prueba. Para cada caso, se sigue el siguiente formato de entrada:

- Una línea que contiene 3 enteros v_t, v_o y d indicando la velocidad del taxi, velocidad del omnibus, y la distancia entre la terminal y el omnimus ($0 \leq v_t, v_o, d$).

Output

Para cada caso de prueba, el programa deberá imprimir el tiempo en el cual el taxi puede alcanzar al omnibus con 10 dígitos de precisión, si fuese imposible entonces deberá imprimir -1.

Example

Standard input	Standard output
2 2 2 2 5 3 11	-1 5.5000000000

Observación

Para imprimir un número con 10 dígitos de precisión:

C `printf("%.10f", valor);`

C++ `std::cout << std::set_precission(10) << valor;`

C# `System.Console.WriteLine("{0:F10}", valor);`

Python `print "%.10f" % valor`

Java `System.out.printf("%.10f", valor);`