Problemas Concurso Programación UDEM 2020

Problema A. *Arbolito de navidad*

Entrada Estándar
Tiempo límite: 3 seg.

Emilio es un niño pequeño que le gustan los arbolitos de navidad, y ahora que se acerca la época navideña, quieres hacerle como regalo un programa que dibuje un arbolito de asteriscos por computadora.

El programa deberá pedir un número natural que representa la altura del arbolito y posteriormente deberá dibujar un arbolito con la altura indicada como se muestra a continuación:

Entrada

La entrada es un entero positivo N, donde $0 < N \leq 50$.

Salida

Un arbolito de asteriscos con altura N como se muestra en los ejemplos.

Ejemplos de casos:

```
Dame la altura: 1
*
```

```
Dame la altura: 7
      *
     ***
    *****
   *******
  *********
 ***********
*************
```

```
Dame la altura: 4
   *
  ***
 *****
*******
```

Problema B. *Contando Estrellas*

Entrada Estándar
Tiempo límite: 3 seg.

En esta ocasión, tienes que escribir un programa que cuente las estrellas que hay en en el cielo. Para este problema, un cielo está representado como una cuadricula finita, en donde cada celda pueda haber dos tipos de caracteres, '.' (punto) que representa al vacío y '*' (asterisco) que representa "algo" en el cielo.

En nuestro cielo las estrellas son los astros pequeños y no pueden ocupar mas de una celda y dos estrellas nunca son adyacentes. Por lo tanto, dos o más caracteres '*' adyacentes indican que algún objeto más grande está en el cielo, como la Luna, un cometa, el Sol, un planeta, etc.

Se consideran celdas adyacentes las ocho posibles celdas alrededor de una celda. Por ejemplo, en la siguiente figura se observa que la celda en el centro tiene ocho celdas adyacentes.

*	.	.
.	*	*
.	.	*

Entrada

En la entrada se te darán varios casos de prueba que representan un cielo en un momento particular. Por cada cielo, se proporciona primero una línea con dos números enteros f y c (separados por un espacio), cuyos valores pueden estar en un rango de 1 a 100, que indican el número de filas y columnas respectivamente del tamaño del cielo. Enseguida estarán f líneas que describen al cielo como se indicó anteriormente. La entrada termina cuando se proporciona una línea que contenga dos ceros.

Salida

Por cada cielo, tu programa deberá imprimir el número de estrellas que contiene.

Ejemplo de Entrada

```
5 5
.*...
....*
....*
...*.
*....
4 3
...
.*.
...
*.*
1 1
*
0 0
```

Ejemplo de Salida

```
2
3
1
```

Problema C. *Distancia Recorrida*

Entrada Estándar
Tiempo límite: 3 seg.

Imagina que viajas en carretera y quieres saber cuanta distancia has recorrido, pero notas que el odómetro de tu vehículo se ha descompuesto, afortunadamente el velocímetro y el control de crucero si funcionan. Entonces el automóvil puede mantener una velocidad constante que puedes ir ajustando de vez en cuando para respetar los límites de velocidad, atascos, etc.

Tu copiloto trae consigo un cronómetro, por lo que puede anotar el tiempo transcurrido cada vez que cambias de velocidad con el control crucero. Escribe un programa que ayude a resolver el problema de saber la distancia recorrida.

Entrada

La entrada tendrá varias líneas de datos: cada cambio de velocidad se indica mediante una línea que especifica el tiempo transcurrido desde el inicio del viaje en formato **hh:mm:ss**, seguido de la nueva velocidad en km/h. Cada distancia que se desea conocer está indicada por una línea que contendrá el tiempo transcurrido en el mismo formato hh:mm:ss. Asume que al comienzo del viaje, el automóvil está parado y que los tiempos transcurridos siempre se dan en orden creciente y que al menos hay un cambio de velocidad en la entrada. También que el vehículo no supera los 200 km/h y que el tiempo transcurrido máximo será 99:59:59

Salida

Para cada consulta de distancia transcurrida, tu programa debe imprimir una línea con el tiempo y la distancia recorrida en kilómetros.

Ejemplo de Entrada

```
00:00:01 100
00:15:01
00:30:01
01:00:01 50
03:00:01
03:00:05 140
```

Ejemplo de Salida

```
00:15:01 25
00:30:01 50
03:00:01 200
```

Problema D. *Encuentra el Palíndromo*

Entrada Estándar
Tiempo límite: 3 seg.

Existe un método simple llamado **invierte y suma**, que consiste en tomar un número entero, invertir sus dígitos y sumarlo al original. Si la suma no es un palíndromo (es decir, no es el mismo número leído de izquierda a derecha y de derecha a izquierda), el proceso se repite. Por ejemplo:

```
  354  <--número inicial
+ 453
  ---
  807
+ 708
 ----
 1515
+5151
 ----
 6666  <-- Palíndromo
```

En este ejemplo, el palíndromo "6666" apareció después de la tercer suma. Este método genera palíndromos en unos pocos pasos para "casi" todos los números enteros. Pero existen excepciones como el 196 que hasta la fecha no se ha encontrado ningún palíndromo con este método.

Escribe un programa que genere un palíndromo y cuente el número de sumas para encontrar dicho palíndromo. Para cada uno de los casos para validar este problema asume que:

- Todos tienen respuesta.
- Se obtiene en menos de 1000 sumas.
- El palíndromo producido no será mayor a 4,000,000,000

Entrada

La primera línea de la entrada será un número n ($0 < n \leq 100$) con el número de casos de prueba, las siguientes n líneas tendrán un número x del cual se desea buscar el palíndromo.

Salida

Para cada una de las n pruebas imprime una línea con el número de sumas para llegar al palíndromo seguido del palíndromo resultante, separados por un espacio.

Ejemplo de Entrada

```
3
354
170
195
```

Ejemplo de Salida

```
3 6666
2 383
4 9339
```

Problema E. *Pepe Pecas*

Entrada Estándar
Tiempo límite: 3 seg.

Los **tautogramas** son un caso especial de aliteración, que consiste en la aparición de la misma letra al comienzo de palabras adyacentes. Entonces, una oración es un tautograma si todas sus palabras comienzan con la misma letra. Por ejemplo, las siguientes oraciones son tautogramas:

- Pepe Pecas pica papas
- Antes alegre andaba
- doce docenas de duraznos

Escribe un programa que dada una serie de frases, compruebe si son un tautograma o no.

Entrada

Cada caso de prueba vendrá en una línea que contendrá una oración. Una oración está formada por una secuencia de entre 1 a 50 palabras separadas por espacios simples. Una palabra está formada por una secuencia de entre 1 a 50 letras mayúsculas y minúsculas (únicamente a-z, A-Z), considera que no aparecerán acentos ni eñes.

Los casos terminan con una línea que contendrá únicamente el carácter de '*' (asterisco).

Salida

Para cada caso de prueba, imprime una sola línea que contenga la palabra "SI" si la oración es un tautograma, o la palabra "NO" en caso contrario.

Ejemplo de Entrada

```
Pepe Pecas pica papas
Antes alegre andaba
doce docenas de duraznos
flores de Francia
esto no es un tautograma
*
```

Ejemplo de Salida

```
SI
SI
SI
NO
NO
```

Problema F. *Recuperando Archivos*

Entrada Estándar
Tiempo límite: 3 seg.

Una persona ha intentado destruir evidencia de un archivo importante de una computadora, la policía ha detectado que el archivo no fue destruido del todo, y pudo ser recuperado, pero no como se esperaba.

La policía logró obtener varias copias del archivo, pero lamentablemente cada copia está fragmentada en 2 pedazos y no se sabe que parte corresponde a cual. Se han guardado todos los pedazos del archivo y te han llamado para pedirte ayuda para recuperar el archivo original.

Puedes estar seguro que cada par de pedazos correctamente unidos formarán el archivo original, y que todos los pedazos proporcionados tienen pareja. El equipo de la policía ya ha traducido por ti los fragmentos binarios a cadenas de 1's y 0's, y te pide que escribas un programa para determinar el patrón de bits que contiene el archivo original.

Entrada

La entrada comienza con un entero positivo n que indica el número de casos, cada caso se describe a continuación: La primer línea indica el número de pedazos p, $2 \leq p \leq 200$ (siempre será par) del archivo recuperado. Enseguida estarán p líneas de 1's y 0's que representan cada pedazo de archivo recuperado. Puedes asumir que cada pedazo no es mayor a 64 bytes.

Salida

Para cada caso de prueba, imprime una sola línea de 1's y 0's que representan al patrón de bits del archivo original. En el caso de que exista más de una solución, puedes imprimir cualquiera de las posibles soluciones.

Ejemplo de Entrada

```
2
6
011
0111
01110
111
0111
10111
4
1
0000
1111
1110000
```

Ejemplo de Salida

```
01110111
11110000
```

Problema G. *Suma de Primos*

Entrada Estándar
Tiempo límite: 3 seg.

Un número primo es un número entero mayor que 1 cuyos divisores positivos son él mismo y el 1. El número 1, por convenio, NO se considera primo.

Te piden hacer un programa que indique si un número N puede expresarse como la suma de dos números primos. En caso de que N no se puede expresar con dos números primos deberás imprimir el mensaje "No se puede". En caso de que si exista solución debes imprimir los dos números primos que den la suma de N, si hubiera más de una solución, imprime cualquier par. Puedes asumir que N no será nunca mayor a 1000.

Entrada

La entrada es un entero positivo N.

Salida

El par de números primos, separados por un espacio, o la frase "No se puede".

Ejemplos de casos:

16 3 13	8 5 3	17 No se puede