NOTA: Si usted está leyendo este documento sin haber extraído el compactado que se le entregó, ciérrelo ahora, extraiga todos los archivos en el escritorio, y siga trabajando desde ahí. Es un error común trabajar en la solución dentro del compactado, lo cual provoca que los cambios no se guarden. Si usted comete este error y entrega una solución vacía, no tendrá oportunidad de reclamar.

Después de pasar la escuela para convertirse en heroe, Fito se enfrenta a sus primeros días de trabajo ejerciendo la profesión. Tiene que cumplir una serie de misiones con diferentes grados de dificultad en varios días consecutivos (una misión cada día). Fito tiene a su disposición tres ayudantes que enumeraremos 0, 1 y 2. En cada misión solo pueden participar dos ayudantes. Cada ayudante seleccionado recibe una bonificación en dependencia de la misión en la cual participa. Se tiene una tabla que define la bonificación que recibe cada ayudante según la misión. El problema consiste an ayudar a Fito a seleccionar dos ayudantes para cada misión de manera tal que al final cada ayudante haya acumulado la misma bonificación y que a su vez esta sea la mayor posible.

	Ayudante 0	Ayudante 1	Ayudante 2
Misión 1	1	1	2
Misión 2	1	0	2
Misión 3	1	3	1
Misión 4	1	0	2

La figura anterior muestra una tabla de las bonificaciones posibles a otorgar para 4 misiones. Si se hace la siguiente selección

```
Misión 1:  A0 y A2
Misión 2:  A0 y A1
Misión 3:  A1 y A2
Misión 4:  A0 y A1
```

Se obtiene que cada ayudante acumularía un total de 3 en bonificaciones. Pero si se hiciese la selección

```
Misión 1:  A0 y A1
Misión 2:  A0 y A2
Misión 3:  A0 y A1
Misión 4:  A0 y A2
```

Se obtendría un acumulado igual para cada uno pero de 4, lo que es una mejor solución porque el acumulado es mayor.
Debe notarse que puede haber una tabla de misiones-bonificaciones para las que no haya solución, es decir, que no existe ninguna selección posible de ayudantes que de como resultado final, un acumulado total de bonificaciones que sea igual para cada ayudante.(ver ejemplo 2)

Usted debe implementar el método `IEnumerable<Tuple<int, int>> Resuelve(int[,] bonificaciones)`. El método recibe un array de `m` filas y `n` columnas donde la posición `i, j` representa la bonificación que recibe el ayudante `j` en la misión `i`. Usted debe retornar un `IEnumerable<Tuple<int, int>>` donde la k-ésima tupla tiene como elementos los dos ayudantes seleccionados para la misión k.
En caso de haber más de una solución basta con retornar una cualquiera de ellas. Si no hubiese solución debe retornarse un `IEnumerable<Tuple<int, int>>` vacío (sin elementos).
Como se ha dicho la cantidad de ayudantes (es decir la cantidad n de columnas) es 3 y para simplificar la cantidad de misiones (la cantidad m de filas) siempre será un valor menor que 16.
El tipo `Tuple<int, int>` representa un par de ayudantes, es decir que `new Tuple<int, int>(0, 2)` representaría al par de ayudantes 0 y 2.

Ejemplo 1:

```
int[,] datos0 =
{
    {1, 0, 0},
    {0, 1, 0},
    {0, 0, 1}
};
foreach (var x in Resuelve(datos0))
  Console.WriteLine(x.Item1 + " " + x.Item2);
 /*
    Imprime:
    0 2
    1 2
    1 2
 */
```

En este ejemplo se seleccionan para la primera misión los ayudantes 0 y 2, para la segunda misión los ayudantes 1 y 2 y para la tarcera los ayudantes 1 y 2. Note que la máxima bonificación acumulada igual para los tres es 1. Otra salida válida que también daría un máximo acumulado igual de 1 sería el enumerable `{0 1}, {1 2}, {0 2}`.

Ejemplo 2:

```
int[,] datos1 =
{
    {1, 0, 0},
    {1, 1, 0},
};
foreach (var x in Resuelve(datos1))
  Console.WriteLine(x.Item1 + " " + x.Item2);
 /*
    Imprime:
  */
```

En este ejemplo no existe ninguna selección de los ayudantes que cumpla que el acumulado para las dos misiones sea igual para los tres ayudantes y por tanto el enumerable es vacío y no se imprime nada.

Ejemplo 3:

```
int[,] datos2 =
{
    {0, 0, 0},
};
foreach (var x in Resuelve(datos2))
  Console.WriteLine(x.Item1 + " " + x.Item2);
/*
   Imprime:
   1 2
*/
```

Habría en este caso otros dos posibles resultados válidos el enumerable formado por el tuplo `{0 1}` y el enumerable formado por el tuplo `{0 2}`.

Ejemplo 4:

```
int[,] datos3 =
{
    {0, 0, 1},
};
foreach (var x in Resuelve(datos3))
  Console.WriteLine(x.Item1 + " " + x.Item2);
/*
   Imprime:
   0 1
*/
```

En este ejemplo la única salida es el tuplo `{0, 1}`. Note que el ayudante que no se selecciona mantiene su acumulado igual `0` porque no suma y los dos que se seleccionan tienen bonificación `0`, lo que hace que el acumulado sea igual a `0` para los tres ayudantes.

Ejemplo 5:

```
int[,] datos4 =
{
    {10, 10, 10},
    {10, 10, 10},
    {10, 10, 10}
};
foreach (var x in Resuelve(datos4))
  Console.WriteLine(x.Item1 + " " + x.Item2);
/*
   Imprime:
   0 1
   0 2
   1 2
*/
```

Note que en este ejemplo hay varias soluciones con el acumulado máximo igual a `20`.

NOTA: Los casos de prueba que aparecen en este proyecto son solamente de ejemplo. Que usted obtenga resultados correctos con estos casos no es garantía de que su solución sea correcta y de buenos resultados con otros ejemplos. De modo que usted debe probar con todos los casos que considere convenientes para comprobar la validez de su implementación.