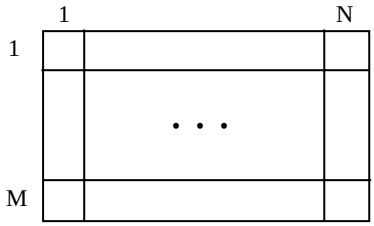
TOM Y JERRY

Como es usual, el gato Tom persigue a Jerry por las habitaciones de una casa sin ningún éxito, pues el maldito roedor conoce todos los recovecos de la mansión. Harto de esta situación, Tom adquiere un PC con la intención de construir un programa capaz de encontrar el camino más corto hasta su ansiada presa sorteando todos los obstáculos (paredes, armarios, etc.) que pueda encontrarse. Además, para asegurar el éxito de la cacería, Tom desea atacar a Jerry cuando éste duerma.

Representaremos la casa por una superficie rectangular de *M* filas y *N* columnas, dividida en casillas unitarias:



Tom tan solo puede moverse en horizontal o vertical dentro de la habitación, nunca en diagonal. Supondremos que tanto Tom como Jerry ocupan una casilla dentro de esta superficie. Los obstáculos de la habitación serán rectángulos de coordenadas válidas dentro de la superficie, y se representarán con su vértice superior izquierdo y el inferior derecho. Existe la posibilidad que el rectángulo sea en realidad una recta (horizontal o vertical; por ejemplo, las coordenadas (3, 4) y (3, 6) definen una recta horizontal) o incluso un punto (cuando las dos coordenadas son iguales).

1.- Construir un programa que procese un fichero de entrada, de nombre "TOM1.IN", que contenga la configuración inicial de la casa, y que la dibuje si es correcta. Los casos de error que consideramos serán: (E0) *M* o *N* (o ambas) son igual a 0. (E1) Tom o Jerry no se encuentran en coordenadas válidas de la superficie de la casa. (E2) Tom y Jerry están en la misma casilla. (E3) Algún obstáculo no está situado en coordenadas válidas de la superficie. (E4) Los vértices que representan un obstáculo no cumplen una relación válida entre sí. (E5) Dos o más obstáculos se solapan. (E6) Tom o Jerry están en una casilla ocupada por un obstáculo.

El formato del fichero "TOM1.IN" es el siguiente:

- Línea 1: valores de *M* y de *N*, separados por un único espacio en blanco (tanto *M* como *N* se representan con un único dígito)
- Línea 2: cuatro valores naturales, separados por un único espacio en blanco, representados cada uno de ellos por un dígito. Los dos primeros valores representan la posición inicial de Tom (fila - columna) y los dos siguientes la de Jerry.
- Líneas siguientes: cada línea se corresponde a un obstáculo y contiene cuatro valores naturales, separados por un único espacio en blanco, y representados cada uno de ellos por un dígito. Los dos primeros valores representan el vértice superior izquierdo del obstáculo (fila - columna) y los dos restantes el vértice inferior derecho.

La salida debe almacenarse en el fichero "TOM1.OUT". En caso de encontrarse

algún error, el fichero contendrá una única línea con el mensaje "ERROR Ex", siendo la 'x' un número entre 0 y 6 que identifica el tipo del primer error que se encuentre en la entrada. Si hubiese más de un error a la vez, se devolverá aquel con código menor; por ejemplo, si el primer error lo provoca un obstáculo que se solapa con alguno de los anteriores y al mismo tiempo ocupa la casilla de Tom o de Jerry, el código a devolver sería E5. En caso de no encontrarse ningún error, el fichero contendrá *M* líneas de *N* caracteres cada una de ellas, siendo cada carácter el contenido de una casilla, que será 'o' para las casillas vacías, 'x' para las casillas que forman parte de un obstáculo, 'T' para la casilla ocupada por Tom y 'J' para la casilla ocupada por Jerry.

A continuación, ofrecemos algunos ejemplos de entrada errónea y el mensaje que debe guardarse en el fichero de salida:

Ejemplos:

TOM1.IN	**TOM1.OUT**
3 3 2 2 3 3 1 2 2 1	ERROR E4
3 3 2 2 3 2 1 2 2 3	ERROR E6
3 3 3 2 2 3 1 1 2 2 1 2 2 3	ERROR E5

En cambio, las entradas siguientes generan resultados correctos:

Ejemplos:

TOM1.IN	**TOM1.OUT**
5 5 1 1 5 3 2 1 3 3 1 4 4 4	Tooxo xxxxo xxxxo oooxo ooJoo
7 5 1 2 7 1 2 1 2 2 2 4 2 4 4 2 4 5 6 1 6 2	oTooo xxoxo ooooo oxxxx ooooo xxoxo Joooo

2.- Construir un programa que, para una entrada que no contenga ningún error, devuelva un camino lo más corto posible que lleve de Tom a Jerry. El programa leerá la información del fichero "TOM2.IN", con el mismo formato que el fichero "TOM1.IN" del apartado anterior. La salida se almacenará en el fichero "TOM2.OUT", que contendrá diferentes líneas con el resultado del algoritmo. En concreto, el fichero contendrá una línea para cada posición que forme parte del camino en el mismo orden en que aparecen en él, siendo pues la primera la posición de Tom y la última la posición de Jerry. Concretamente, cada línea contendrá un par de valores naturales, representados mediante un dígito y separados por un único espacio en blanco; el primer natural representa la fila y el segundo la columna. En caso de que no haya ninguna solución posible, el

fichero "TOM2.OUT" deberá contener una única línea con la palabra "INALCANZABLE". En caso de que haya más de un camino mínimo, podéis devolver cualquiera de ellos.

Así para las entradas del último ejemplo del apartado anterior, para el primero de ellos el fichero "TOM2.OUT" debería contener una única línea con la palabra "INALCANZABLE". En cambio, para el segundo, al existir un único camino mínimo, el resultado sería el fichero "TOM2.OUT" siguiente:

Ejemplo:

TOM2.IN	**TOM2.OUT**
4 4 1 1 4 4 3 2 3 3 2 3 2 3	1 2 1 3 2 3 3 3 3 2 3 1 4 1 5 1 5 2 5 3 6 3 7 3 7 2 7 1

3.- Construir un programa que, para una entrada que no contenga ningún error, devuelva todos los caminos que lleven de Tom a Jerry, sean lo largos que sean, y tales que no se pasa más de una vez por una misma casilla. Para minimizar el tamaño de la salida, nos limitaremos a decir cuántos caminos hay de cada longitud. La longitud de un camino se define como el número de posiciones que lo componen menos uno; en el último ejemplo del apartado anterior, el camino que aparece tiene longitud 13.

El programa leerá la información del fichero "TOM3.IN", con el mismo formato que el fichero "TOM2.IN" del apartado anterior. La salida se almacenará en el fichero "TOM3.OUT", que contendrá diferentes líneas. Cada una de las líneas estará formada por dos valores naturales, representados mediante tantos dígitos como sea necesario y separados por un único espacio en blanco; el primer natural representa la longitud y el segundo el número de caminos existentes de esa longitud. No se debe incluir líneas para aquellas longitudes que no sean longitudes de algún camino de Tom a Jerry. Las líneas deben estar ordenadas por la longitud.

La figura siguiente muestra un ejemplo de entrada y el resultado esperado para ella.

Ejemplo:

TOM3.IN	**TOM3.OUT**
4 4 1 1 4 4 3 2 3 3 2 3 2 3	6 2 8 2