Galopada

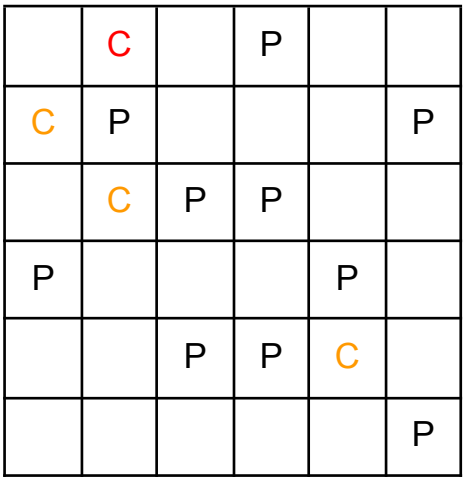
Sobre un tablero de ajedrez se distribuyen una serie de peones. Se trata de encontrar una de las casillas libres del tablero desde la que un caballo pueda realizar una “galopada”; es decir, comerse todos los peones en una sola secuencia de movimientos, sin pisar ninguna otra de las casillas vacías.

Defina en OCaml, utilizando solo recursos propios de la programación funcional, una función

galopada: int -> (int * int) list -> (int * int) list

de modo que ***galopada n peones*** encuentre esa casilla (si existe) y devuelva una lista cuya cabeza sea las coordenadas de esa casilla y su cola contenga las coordenadas de los peones en el orden en que serían comidos en la galopada. ***n*** sería la dimensión del tablero (filas y columnas estarían numeradas de *1* a *n)* y ***peones*** sería la lista con las coordenadas de todos los peones.

La función ***galopada*** debe provocar la excepción ***Not_found*** si el problema no tiene solución y la excepción ***Invalid_argument “galopada”*** si alguno de los argumentos no es válido (la dimensión debe ser mayor o igual que *1* y ***peones*** no debe contener elementos repetidos y todos ellos deben corresponder a coordenadas válidas en el tablero de dimensión ***n***)




```
# galopada 6 [(1,4);(2,6);(3,4);(4,5);(2,2);(3,3);(6,6);(5,4);(5,3);(4,1)];;
- : (int * int) list =
[(1,2);(3,3);(1,4);(2,6);(3,4);(2,2);(4,1);(5,3);(4,5);(6,6);(5,4)]
```

Tenga en cuenta que su función no tiene por qué dar necesariamente esa misma solución. Otra posibilidades serían, por ejemplo:

```
[(2,1);(3,3);(1,4);(2,6);(3,4);(2,2);(4,1);(5,3);(4,5);(6,6);(5,4)]
```

o

```
[(5,5);(3,4);(2,6);(1,4);(2,2);(4,1);(3,3);(5,4);(6,6);(4,5);(5,3)]
```

o

```
[(3,2);(5,3);(3,4);(2,6);(4,5);(6,6);(5,4);(3,3);(1,4);(2,2);(4,1)]
```