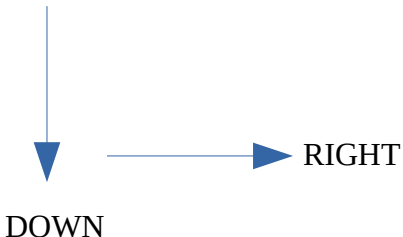


Exámen de Laboratorio de Algoritmos y Estructuras de Datos II

El TAD *position* representa una celda en una matriz cuadrada de dimensión N.

	0	1	2	3	4	5
0	□	□	□	□	□	□
1	□	□	□	□	□	□
2	□	□	□	□	□	□
3	□	[q]	□	□	□	□
4	□	□	□	□	□	□
5	□	□	□	□	□	□



position_initial(6) => Posición inicial en una matriz de 6x6, igual a la celda (0, 0)

position_left(p) => Mueve la posición *p* una columna hacia la izquierda

position_right(p) => Mueve la posición *p* una columna hacia la derecha

position_up(p) => Mueve la posición *p* una fila hacia arriba

position_down(p) => Mueve la posición *p* una fila hacia abajo (incrementa el número de fila)

position_absolute(row, col, dim) => Crea una posición nueva a partir de números naturales.

La posición **q** = (3, 1) del tablero anterior se puede construir de la siguiente manera:

q = *position_down* (*position_down* (*position_down* (*position_right* (*position_initial* (6)))))

Todos los movimientos son circulares, por ejemplo, en una matriz 6x6,

position_right((1, 5)) = (1, 0) ya que no es posible moverse hacia la derecha.

position_board(p) => Construye una matriz con el valor 0 en todas las celdas, excepto por la celda correspondiente a la posición *p* que tiene valor 1. Por ejemplo, *position_board(q)* devuelve la matriz:

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	1	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0

position_movements(p) => Devuelve una lista de movimientos necesarios para alcanzar la posición *p* a partir de la posición inicial (0,0). La lista contiene una secuencia de movimientos de columnas (hacia la derecha) seguida de una secuencia de movimientos de fila (hacia abajo).

position_movements(q) = [R, D, D, D] *position_movements(position_initial(6))* = []

Ejercicio 1:

En el archivo *positions.c* implementar todas las funciones anteriores utilizando punteros a estructuras. Ningún otro archivo debe modificarse.

Ejercicio 2:

En el archivo *positions.c* se define una función adicional *position_movements_length(p)*. Dicha función retorna la cantidad de movimientos que son necesarios para mover de la posición (0,0) a la posición deseada p. Implemente esa función.

```
position_movements_length(position_initial(6)) = 0  
position_movements_length(q) = 4
```

Ejercicio 3:

En el archivo *main.c* implementar una función *main* que utilice todas las funciones anteriores mostrando al menos un ejemplo. Debe mostrar en pantalla la matriz generada por *position_board* y la lista generada por *position_movements*.

La presencia de memory leaks o invalid reads restará puntos.

Correr tests: \$> make run_tests Correr Main con Valgrind: \$> make run_main