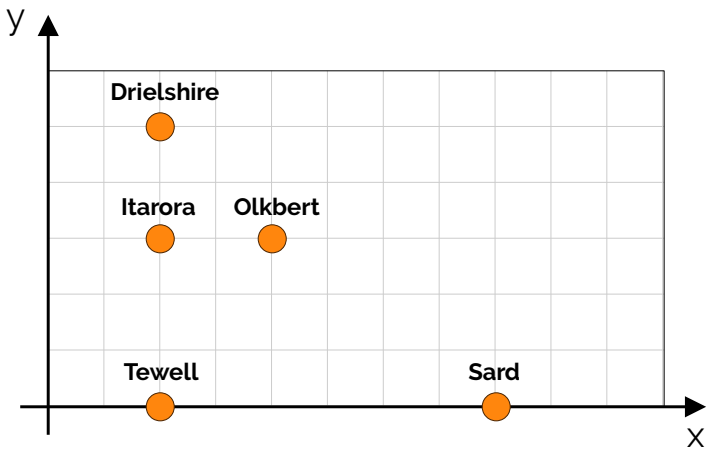
Las torres del desierto

El desierto de Clibrigde alberga numerosas ruinas de torres de varios siglos de antigüedad. Nada se conoce sobre la procedencia de esas torres, pero sí se sabe que sus habitantes utilizaban el sol y la estrella polar para orientarse, ya que algunas de las torres se encontraban alineadas de norte a sur y otras tantas de este a oeste.

Tu labor como cartógrafo del desierto es construir un plano con la situación de cada una de las torres. Utilizarás coordenadas cartesianas para representar la posición de cada una de ellas, tomando como origen de coordenadas la esquina suroeste del desierto. El eje de las abscisas (x) apunta en dirección este y el de las ordenadas (y) apunta en dirección norte.



Se pide implementar un TAD `Desierto` con las siguientes operaciones:

- `anyadir_torre(const string &nombre, int x, int y)`

  Añade una torre al mapa, con el `nombre` y las coordenadas (`x`, `y`) pasadas como parámetro. Si ya existía una torre con el mismo nombre, debe lanzarse una excepción de tipo `domain_error` con el mensaje `Torre ya existente`. Si ya existía una torre en la posición (`x`, `y`), se lanzará una excepción `domain_error` con el mensaje `Posicion ocupada`.

- `eliminar_torre(const string &nombre)`

  Elimina del mapa la torre cuyo nombre coincide con el pasado como parámetro. Si no hay una torre con ese nombre, se lanzará una excepción `domain_error` con el mensaje `Torre no existente`.

- `posicion(const string &nombre) const`

  Devuelve un `pair<int, int>` con las coordenadas de la torre cuyo nombre es el pasado como parámetro. Si no hay una torre con ese nombre, se lanzará una excepción `domain_error` con el mensaje `Torre no existente`.

- `torre_en_posicion(int x, int y) const`

  Comprueba si hay una torre en las coordenadas (`x`, `y`) pasadas como parámetro. Devuelve un `pair<bool, string>` como resultado. La primera componente del par tiene valor `true` si existe

una torre con esas coordenadas, o `false` en caso contrario. Si existe una torre en esas coordenadas, la segunda componente contiene el nombre esa torre. Si no existe, el valor de la segunda componente del par es irrelevante.

- `torre_mas_cercana(const string &nombre, const Direccion &dir) const`

  Devuelve un `string` con el nombre de la torre que nos encontraríamos si andásemos en la dirección `dir` dada partiendo de la torre con `nombre` pasado como parámetro. El tipo de datos `Direccion` está definido del siguiente modo:

  ```
  enum class Direccion { Norte, Sur, Este, Oeste };
  ```

  Si no existe ninguna torre con el `nombre` pasado como parámetro, se deberá lanzar una excepción `domain_error` con el mensaje `Torre no existente`. Si no hay ninguna torre en la dirección dada, se lanzará una excepción `domain_error` con el mensaje `No hay torres en esa direccion`.

Para realizar este ejercicio ten en cuenta que:

1. Has de indicar el coste de cada una de las operaciones.
2. Has de utilizar las clases de la STL de C++.
3. Ninguno de los métodos del TAD debe realizar operaciones de E/S. El manejo de E/S debe hacerse en la función `tratar_caso()`.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso está formado por una serie de líneas, en las que se muestran las operaciones a llevar a cabo, una por cada línea: el nombre de la operación seguido de sus argumentos. La palabra `FIN` en una línea indica el final de cada caso.

Todos los nombres de torres son secuencias de caracteres sin espacios. Las coordenadas están formadas por números entre `0` y `10000`.

Salida

Las operaciones `anyadir_torre` y `eliminar_torre` no producen salida, salvo en caso de error. El resto de operaciones, en caso de éxito, producen una línea cuyo contenido depende de la operación:

- Tras la llamada a `posicion` debe imprimirse una línea con las coordenadas `x y` devueltas, separadas por un espacio.
- Tras la llamada a `torre_en_posicion`, si el par devuelto contiene `false` en la primera componente, debe imprimirse la palabra `NO`. En caso contrario, debe imprimirse la palabra `SI` seguida de la segunda componente del par.
- Tras la llamada a `torre_mas_cercana` debe imprimirse el nombre de la torre devuelto por el método.

Si una operación produce una excepción, debe escribirse una línea con el mensaje de la excepción, y no escribirse nada más para esa operación.

Al final de cada caso de prueba ha de imprimirse una línea con tres guiones (`---`).

Entrada de ejemplo

```
anyadir_torre Tewell 2 0
anyadir_torre Itarora 2 3
anyadir_torre Drielshire 2 5
torre_en_posicion 2 0
torre_en_posicion 1 5
anyadir_torre Olkbert 4 3
anyadir_torre Sard 8 0
posicion Olkbert
torre_mas_cercana Itarora Sur
torre_mas_cercana Itarora Oeste
torre_mas_cercana Tewell Norte
eliminar_torre Itarora
torre_mas_cercana Tewell Norte
FIN
anyadir_torre Qiver 0 0
anyadir_torre Yhin 0 0
anyadir_torre Yhin 0 1
anyadir_torre Qiver 1 3
eliminar_torre Yhin
posicion Inewood
posicion Yhin
torre_mas_cercana Qiver Norte
FIN
```

Salida de ejemplo

```
SI Tewell
NO
4 3
Tewell
No hay torres en esa direccion
Itarora
Drielshire
---
Posicion ocupada
Torre ya existente
Torre no existente
Torre no existente
No hay torres en esa direccion
---
```

Autor

Manuel Montenegro