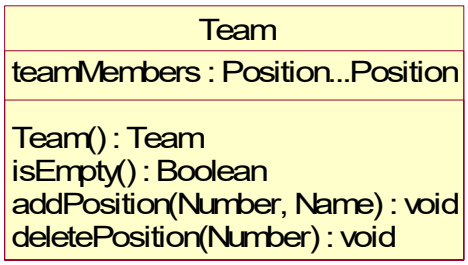
Lab 7: Linked Lists

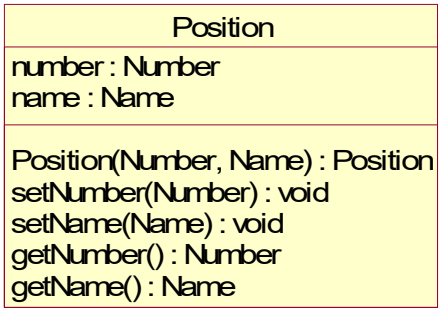
Aim
This lab class will provide you with an opportunity to:

- consolidate your conceptual understanding of linked lists and the two types which make them work (the 'way in' and the node);
- develop an implementation of a node type comprising three fields; and
- write code to manipulate a collection implemented using a linked list.

Context
A *netball team* consists of seven positions: 1 Goal Keeper, 2 Goal Defence, 3 Wing Defence, 4 Centre, 5 Wing Attack, 6 Goal Attack, and 7 Goal Shooter. The relevant portion of the UML diagram for a **Team** which shows it as comprising of a collection of **Position**s is:



A **Position** consists of a position number and a position name. This could be represented as:



This tutorial relates to linked lists. For information-hiding purposes, the `team` type needs to be a pointer to a structure. The structure needs to contain a 'way in' to the linked list. The required types are:

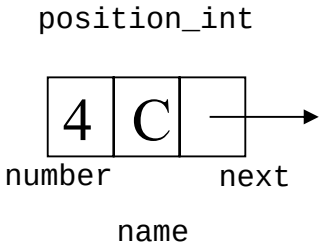
```
struct team_int;
typedef struct team_int *team;
struct team_int
{
    position firstPos;
};
```

It would be possible to implement a `node` in the linked list in which the `data` variable is of type `position` — which would match the simplest form of the node and match what has been introduced in lectures. For the sake of the exercise, however, we will

combine the position and the node together so that the type `position` is used to represent nodes in the linked list and will contain two 'data' fields: a `number` and a `name`.

Tasks

1. Write code to implement the `position` and `struct position_int` types. position should be a pointer to a `struct position_int` which in turn should be defined to implement the following diagram:



2. Write the initialiser function ('constructor') for the `position`. It should have the following function header:

```
void init_position(position *pp, int num, char *name);
```

   and should assign the two parameter values to their corresponding fields and assign `NULL` to `next`.

3. A function, `add_position()`, is required that will create a position with given number and name and then add the position to a team as follows:

```
team t;

init_team(&t);

add_position(t,1,"GK");
add_position(t,2,"GD");
add_position(t,3,"WD");
add_position(t,4,"C");
add_position(t,5,"WA");
add_position(t,6,"GA");
add_position(t,7,"GS");
```

   Write the definition of `add_position()`. You may assume that all positions will be added to the end of the linked list and that the positions will be presented in order of number. Verify that your solution works.

   How would your solution change if you could not assume the positions were presented in order?

4. A function `delete_position()` is required that will delete the position of given number (if it exists) from the given team. Write the definition of `delete_position()`. If a position of given number does not exist, do not report an error, instead simply return the given team unaltered. You may still assume that all nodes are in increasing order of number. Again, verify that your solution works.