**Directed series N° 4**

Exersice 4: (solution)

**Exercise 3**

1. **Sum** function which returns the sum of the elements of a linked list:

**function Sum** (L: List): integer

 **s:integer**

**Begin**

**If** (L=Null ) **then**

 **S🡨 0;**

**Else**

S🡨 L -> ele + Sum (L -> next) ;

**End if**

**Return (s) ;**

**END ;**

1. **Max** function which returns the maximum of a linked list (we will not deal with the case of the empty list)

**function Max** (L: List): integer

 **M : integer;**

**Begin**

**If** (L -> next =Null ) **then**

M🡨 L -> ele;

**Else**

 M 🡨 max(L -> next);

 **If (**L -> ele > M **)then**

 M🡨L -> ele;

 **End if**

**End if**

Return(M);

**END**

1. The function **belongs** to check if an element exists in a list or not.

**Function** belongs (x: int, L: List): Boolean;

 **B: Boolean;**

**Begin**

**If** (L=Null ) **then**

 B 🡨 false;

**Else**

**if** (L -> ele = x) **then**

 B 🡨 true;

**Else**

 B 🡨belongs (x, L -> next);

**End if**

**End if**

Return (B);

**END;**

1. **nb\_occurrences** function allows you to count the number of occurrences of a value x.

**Function** nb\_occurrences (x: integer, L: List): integer;

**Begin**

**If** (L=Null ) **then**

return(0);

**Else**

**if** L -> ele = x **then**

 Return( 1+ nb\_occurrences (x, L -> next)));

**Else**

 Return( nb\_occurrences (x, L -> next)));

**End if**

**End if**

**END ;**