ADT_DCLL: DoubleCircularLinkedList Implementatie datastructuren college3

Implementatie voor Data type Key (Integers)

Type:

```
Key = Integer;
DCLL = ^DoubleCircularLinkedList;
DoubleCircularLinkedList = record
                              HeadKnoop:KnoopPointer;
                              CurrentKnoop: KnoopPointer
                           end;
KnoopPointer = ^Knoop;
P = Knooppointer;
Knoop = record
           LeftKnoop:KnoopPointer;
           Data_Element: DataType;
           RightKnoop: KnoopPointer
        end;
```

```
Procedure FindFirst(var  DCLL:DoubleCircularLinkedList); # bijwerking interne wijzer
begin
    CurrentKnoop := HeadKnoop
end;
Procedure FindRightOne(var  DCLL:DoubleCircularLinkedList); # bijwerking interne wijzer
begin
    CurrentKnoop := ^CurrentKnoop.RightKnoop
end;
```

Procedure **FindLeftOne**(var DCLL:DoubleCircularLinkedList); # bijwerking interne wijzer

begin

	CurrentKnoop := ^CurrentKnoop.LeftKnoop

end;

Procedure **RetrieveThisOne**(var DCLL:DoubleCircularLinkedList;var elem:DataType);

begin

	elem:=^CurrentKnoop.Data_Element

end;

Procedure **UpdateThisOne(**var DCLL:DoubleCircularLinkedList;var elem:DataType);

begin

	^CurrentKnoop.Data_Element := elem

end;

Function **Empty(var DCLL:DoubleCircularLinkedList):**boolean;

Postconditie: Y als linked list leeg (head_pointer=nil) anders N

Function **Last(var DCLL:DoubleCircularLinkedList)**:boolean;	#is CurrentKnoop laatste in lijst?

Postconditie: Y als next_pointer van CurrentKnoop HeadKnoop is, anders N

Procedure **Create(var DCLL:DoubleCircularLinkedList)**;

Postconditie: een lege linked list wordt aangemaakt; HeadKnoop en CurrentKnoop zijn nil.

Procedure **Delete(var DCLL:DoubleCircularLinkedList)**;

Postconditie: geheugen voor DCLL weer vrijgegeven.

```
Procedure InsertAfterThisOne(var  DCLL:DoubleCircularLinkedList;var elem:DataType);
Var p:KnoopPointer;
begin
        with DCLL^ do begin
                new(p);
                p^.Data_Element:=elem;
                if (HeadKnoop=nil)
                then begin                  # lege lijst, eerste knoop met verwijzingen naar zichzelf
                        HeadKnoop:=p;
                        p^.RightKnoop=p;
                        P^.LeftKnoop=p;
                end
                else    begin
                                CurrentKnoop^.RightKnoop^.LeftKnoop:=p;
                                P^.RightKnoop:=CurrentKnoop^.RightKnoop;
                                CurrentKnoop^.RightKnoop:=p;
                                P^.LeftKnoop:=CurrentKnoop;
                        end;
                        CurrentKnoop:=p;
                end
        end;
        ^p.RightKnoop:=^CurrentKnoop.RightKnoop;
        ^CurrentKnoop.RightKnoop:=p;
        ^p.LeftKnoop:=CurrentKnoop
end;
```

Procedure **InsertBeforeThisOne**(var DCLL:DoubleCircularLinkedList;var elem:DataType);

(analoog aan InsertAfterTisOne spiegelsymmetrisch m.b.t. links tussen de knopen)

Procedure **DeleteThisOne(var DCLL:DoubleCircularLinkedList)**;

```
Var  wasCurrentKnoop : KnoopPointer;
begin
        with DCLL^ do begin
                wasCurrentKnoop:=CurrentKnoop;
                If (CurrentKnoop^.RightKnoop = CurrentKnoop^.LeftKnoop) then begin
                        If (CurrentKnoop^.RightKnoop=CurrentKnoop) then begin
                                HeadKnoop:=nil;
                                CurrentKnoop:=nil
                        end;
                end
                else begin
                        Current^.RightKnoop^.LeftKnoop:=CurrentKnoop^.RightKnoop;
                        Current^.LeftKnoop^.RightKnoop:=CurrentKnoop^.LeftKnoop;
                        CurrentKnoop:=CurrentKnoop^.RightKnoop;
                        If (HeadKnoop=WasCurrentKnoop) then HeadKnoop:=CurrentKnoop;
                end;
                Dispose(wasCurrentKnoop)
        end
end;
```