ADT_SingleLinkedList Complete Specificatie datastructuren college3

Elementen:

Elementen zijn lijst, knopen, en hulpwijzers;

De lijst wijst naar een record met administratieve gegevens rond de lijst met knopen;

elke knoop bevat een data_element en een pointer naar een volgende knoop;

elke hulpwijzer geeft het adres van een knoop;

```
Type: LijstPointer      =   ^SingleLinkedList;
      SingleLinkedList  =   record
                                HeadKnoop:KnoopPointer;
                                CurrentKnoop: KnoopPointer
                            end;
      KnoopPointer      =   ^Knoop;
      Knoop             =   record
                                Data_Element: DataType;
                                NextKnoop: KnoopPointer
                            end;
```

Structuur: relatie is lineair, NextKnoop wijst naar opvolger; NextKnoop laatste knoop is nil.

Twee extra lijst pointers wijzen naar voorste knoop (HeadKnoop) en huidige knoop (CurrentKnoop), nil als lijst leeg.

Domein: het aantal knopen in de lijst is eindig [0..Max) # open einde afhankelijk beschikbaar geheugen

Operaties: een 10 tal; operaties vinden meestal plaats bij of vanaf CurrentKnoop (ThisOne); bij elke operatie moeten wijzigingen in de HeadKnoop en CurrentKnoop worden aangegeven; een nieuwe knoop wordt CurrentKnoop na invoeging.

Procedure **FindFirst**(var SLL:SingleLinkedList); # bijwerking interne wijzer

Postconditie: CurrentKnoop wijst naar voorste knoop (net als HeadKnoop).

Procedure **FindNext**(var SLL:SingleLinkedList); # bijwerking interne wijzer

Preconditie: CurrentKnoop is niet laatste.

Postconditie: de volgende knoop waar de NextKnoop van de CurrentKnoop naar verwijst wordt de CurrentKnoop; als NextKnoop nil blijft CurrentKnoop gelijk.

Procedure **RetrieveThisOne**(var SLL:SingleLinkedList;var elem:DataType);

Preconditie: linked list is niet leeg.

Postconditie: elem heeft als waarde die van Data_Element in CurrentKnoop gekregen.

Procedure **UpdateThisOne(**var SLL:SingleLinkedList;var elem:DataType);

Preconditie: linked list is niet leeg.

Postconditie: Data_Element CurrentKnoop heeft waarde elem gekregen.

Procedure **InsertThisOne**(var SLL:SingleLinkedList;var elem:DataType);

Preconditie: er is nog geheugen voor een nieuwe knoop

Postconditie: een nieuwe knoop met als Date_Element elem en als NextKnoop waarde die van de huidige CurrentKnoop wordt toegevoegd; NextKnoop huidige CurrentPointer wijst naar nieuwe knoop; CurrentKnoop wijst naar deze nieuwe knoop; als HeadKnoop nil was krijgt deze ook de waarde van CurrentKnoop.

Procedure **DeleteThisOne(var SLL:SingleLinkedList)**; # duurste operatie

Preconditie: linked list is niet leeg

Postconditie: CurrentKnoop verwijderd uit de lijst; voorganger's next_pointer moet CurrentKnoop's next pointer overnemen; CurrentKnoop wijst naar deze voorganger, als CurrentKnoop gelijk aan HeadKnoop was, wordt CurrentKnoop's NextKnoop zowel HeadKnoop als CurrentKnoop (eventueel dus beide nil als er maar 1 knoop in de lijst zat).

Function **Empty(var SLL:SingleLinkedList):**boolean;

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

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

Postconditie: Y als next_pointer van CurrentKnoop is nil en CurrentKnoop zelfniet nil, anders N

Procedure **Create(var SLL:SingleLinkedList)**;

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

Procedure **Delete(var SLL:SingleLinkedList)**;

Postconditie: geheugen voor SLL weer vrijgegeven.