Übungsblatt 11

(15. Januar 2009)

Aufgabe 1 (1 Punkt) Gegeben sei folgende Listenspezifikation aus der Vorlesung:

Li.h

```
1
2  typedef void* Object;
3  typedef enum {false,true} boolean;
4
5  struct LiS {
6    Object head;
7    struct LiS* tail;
8    boolean isEmpty;
9  };
10
11 typedef struct LiS Li;
12
13 Li* newNil();
14 Li* newCons(Object hd, Li* tl);
15
16 void deleteLi(Li* this);
17
```

Li.c

```
1  #include "Li.h"
2  #include <stdlib.h>
3  Li* newNil(){
4    Li* this = (Li*)malloc(sizeof(Li));
5    this->isEmpty = true;
6    return this;
7  }
8  Li* newCons(Object hd, Li* tl){
9    Li* this = (Li*)malloc(sizeof(Li));
10   this->isEmpty = false;
11   this->head = hd;
12   this->tail = tl;
13   return this;
14 }
15
16 void deleteLi(Li* this){
```

```
17   if (!this->isEmpty) deleteLi(this->tail);
18   free(this);
19 }
20
```

a) Schreiben Sie unterschiedliche Beispielanwendungen für Listen, in denen sie jeweils für unterschiedliche Typen der Elemente benutzt werden.

b) Ergänzen Sie die Datenstruktur `Li` um die folgenden weiteren Funktionen. Wenn nicht anders angegeben lösen Sie die Funktionen rekursiv.

`int length(Li* this);`
Zur Berechnung der Länge der Liste;

c) `Object last(Li* this);`
Zur Berechnung des letzten Listenelements.

d) `Object getlast(Li* this,int i);`
Zur Berechnung des Listenelements an der i-ten Stelle.

e) `Li* reverse(Li* this,int i);`
Eine neue Liste soll erzeugt werden, in der die Listenelemente in umgekehrter Reihenfolge auftreten. Schreiben Sie diese Funktion iterativ.

Aufgabe 2 Ergänzen Sie die Datenstruktur `Li` um die folgenden weiteren Funktionen. Wenn nicht anders angegeben lösen Sie die Funktionen rekursiv.

a) `boolean containsWithProperty(Li* this,boolean prop(Object));`
Sie soll genau dann wahr sein, wenn die übergebene Testfunktion `prop` für eines der Elemente in der Liste wahr ergibt.

b) `boolean isSorted(Li* this,boolean le(Object,Object));`
Sie soll genau dann wahr sein, wenn alle Elemente mit der übergebenen Vergleichsfunktion `le` kleiner sind als ihr Nachfolgerelement in der Liste.