# Elementare Algorithmen und Objektorientierte Programmierung

Übung 5

Auszuarbeiten bis 22.04.21

## 1. Bibliothek zum Arbeiten mit einfach-verketteten Listen (20 Punkte)

Entwickeln Sie eine C-Bibliothek, die den Umgang mit einfach-verketteten Listen zur Speicherung von Ganzzahlen-Datentypen erlaubt. Realisieren Sie folgende Funktionen

- **`node_t* createNode(value_t value)`** erzeugt einen Knoten, speichert den als Parameter übergebenen Wert und gibt die Adresse des neuen Knotens oder NULL (falls kein Speicher vorhanden ist) als Rückgabewert zurück.
- **`Node_t* getNode(node_t* pHead, int index)`** gibt entweder den Knoten mit dem übergebenen Index `index` oder NULL für ungültige Indexwerte zurück.
- **`int insertFront(node_t** ppHead, const value_t* pValue)`** fügt am Anfang der Liste einen Knoten, der mit `value` initalisiert wurde, ein. Der Rückgabewert dient dazu über Erfolg bzw. mögliche Misserfolge zu informieren.
- **`int insertBefore(node_t** ppHead, node_t* pPrevNode, const value_t* pValue)`** fügt vor dem als `ppPrevNode` übergebenen Knoten einen neuen Knoten, der mit `value` initalisiert wurde, ein. Der Rückgabewert dient dazu über Erfolg bzw. mögliche Misserfolge zu informieren.
- **`int deleteValues(node_t** ppHead, const value_t* pValue, int deleteAllOcc)`** löscht, abhängig vom Flag deleteAllOcc, den ersten oder alle Knoten der verkettenen List mit einem bestimmten Wert. Der Rückgabewert dient dazu über Erfolg bzw. mögliche Misserfolge bei der Zerstörung der Liste zu informieren.
- **`int deleteList(node_t** ppHead)`** löscht alle Knoten der verkettenen List und setzt den Head-Pointer am Ende auf NULL. Der Rückgabewert dient dazu über Erfolg bzw. mögliche Misserfolge bei der Zerstörung der Liste zu informieren.
- **`void printList(const node_t* pHead)`** gibt alle Werte der Liste aus
- **`int inverseList(node_t** ppHead)`** kehrt die mit Hilfe von ppHead übergebene Liste um ohne dabei neue Knoten zu erzeugen oder Werte zu kopieren. Der Rückgabewert dient dazu über Erfolg bzw. mögliche Misserfolge zu informieren.
- **`int sortList(node_t** ppHead)`** sortiert die mit Hilfe von ppHead übergebene Liste aufsteigend ohne dabei neue Knoten zu erzeugen oder Werte zu kopieren. Der Rückgabewert dient dazu über Erfolg bzw. mögliche Misserfolge zu informieren.

Das Testen der Bibliotheken soll vollkommen automatisch erfolgen. Achten Sie bei Ihrer Implementierung auf Robustheit und den korrekten Umgang mit dynamisch angelegtem Speicher

# 1 Bibliothek zum Arbeiten mit einfach-verketteten Listen

Aufgabe war es eine C-Bibliothek zu entwickeln mit der man einfach verkettete Listen erstellen, bearbeiten und löschen kann.

**Node_t* createNode(value_t value**

Hier wird einfach mit dem übergebenen Wert ein neuer node erzeugt und zurückgegeben.

**Node_t* getNode(node_t* pHead, int index)**

Hier wird von einer gegeben Liste eine Node zurück geliefert. Der index gibt an an welcher stelle wir den Node gefunden haben.

**int insertFront(node_t** ppHead, const value_t* pValue)**

HIer wird am Anfang der Liste, mithlfe von pointer umbiegen, ein neu erstellter Node mit einem bestimmten Wert eingefügt.

**int insertBefore(node_t** ppHead, node_t* pPrevNode, const value_t* pValue)**

Hier wird ein Node vor dem **PrevNode** eingefügt. Zuerst wird hier die position des prevNode gesucht. Dannach wird ein neuer Node vor dem **PrevNode** eingefügt.

**int deleteValues(node_t** ppHead, const value_t* pValue, int deleteAllOcc)**

Wenn deleteAllOcc false ist wird der erste Node gesucht bei dem der Wert übereinstimmt. Dieser wird dann raus gelöscht (Speicher wird dabei freigegeben).
Wenn deleteAllOcc True ist werden alle Nodes mit diesem Wert gelöscht (Speicher wird dabei freigegeben).

**int deleteList(node_t** ppHead)**

Hier wird solange über die Liste iteriert (while (*ppHead != NULL)) bis alle gelöscht sind.

**void printList(const node_t* pHead)**

Hier wird über die Liste drüber iteriert (while (pCurr != NULL)) und bei jeder Iteration wird der aktuelle Node ausgegeben

**int inverseList(node_t** ppHead)**

Hier wird die ganze Liste invertiert indem man einfach in einer while Schleife solange die Pointer umbiegt bis pCur == NULL ist.

```
Node_t* pCurr = *ppHead;
Node_t* pPrev = NULL;
Node_t* pNext = NULL;

while (pCurr != NULL) {
    pNext = pCurr->pNext;
    pCurr->pNext = pPrev;

    pPrev = pCurr;
    pCurr = pNext;
}
```

**int sortList(node_t** ppHead)**
Hier werden alle Nodes mit dem Selection sort Algorithmus sortiert.

**Testfälle**:
siehe main.c file.

C-Program code: Siehe beigelegtes linkedList.c/.h file.