TECHNISCHE HOCHSCHULE DEGGENDORF	**Studiengang:** ___AI_____	**Platzziffer:** _____	Punkte:	
			Note:	

Fakultät Elektro- und Medientechnik			
Kurs:	Algorithmen und Datenstrukturen	**Prüfungssemester:**	WS 17/18
		Prüfungsdauer:	90 Min.
Prüfer:	Prof. Dr. Peter Jüttner	**Prüfungsdatum:**	30.1.2018

☞ *Keinen Rotstift verwenden!*	☞ *zugelassene Hilsmittel: Alle*
☞ *nur an den vorgesehenen Stellen antworten!*	☞ *ggf. Rückseiten nutzen!*

Vorbemerkung: include Anweisungen für die Bibliotheken stdlib, stdio, string, math müssen nicht angegeben werden! Funktionen aus diesen Bibliotheken dürfen verwendet werden, sofern nicht anders angegeben

1. O-Notation, Komplexitätsklassen

Warum ist es meist nur theoretisch und nicht praktisch möglich Probleme der Komplexitätsklasse $O(2^n)$ durch ein Computerprogramm zu lösen. (2 Punkte)

a. Wie verändert sich die Laufzeit eines Programms der Komplexitätsklasse $O(2^n)$ tendenziell, wenn die Menge der zu verarbeitenden Daten verdoppelt wird? (3 Punkte)

b. Warum kann ein Problem der Komplexitätsklasse $O(n^2)$ nicht mit einem Algorithmus der Komplexitätsklasse $O(n)$ gelöst werden? Geben Sie **einen** Grund an. (3 Punkte)

2. Aufwand eines Algorithmus zum Bilden des Skalarprodukts zweier Vektoren.

Im Folgenden ist ein Algorithmus (als C-Prozedur) mit drei Parametern v1 und v2 vom Typ int[], n vom Typ unsigned int angegeben, der das Skalarprodukt der Vektoren v1 und v2 ermittelt und als Funktionsergebnis per return zurückgibt. Dabei gibt n die Länge der Vektoren v1 und v2 an, d.h. die Anzahl der Elemente von v1 und v2.
Der Algorithmus geht davon aus, dass n „sinnvoll“, d.h. größer 0 ist.

```
int skalarprodukt (int v1[], int v2[], unsigned int n)
{
  int i;
  int produkt = 0;
  for (i=0; i<n; i++)
  {
     produkt = produkt + v1[i] * v2[i];
  }
  return produkt;
}
```

a. Geben Sie an, wie viele Rechenoperationen (Summe der Additionen und Multiplikationen) für eine Vektorlänge n=4 durchgeführt werden. (6 Punkte)

   Hinweise:
   - Vergessen Sie nicht die Rechenoperation im Schleifenkopf.

b. Erläutern Sie, warum der Aufwand das Problem auf diese Art zu lösen, von der Ordnung O(n) ist. n sei dabei die Länge des Vektors v. (5 Punkte)

c. Ist der Aufwand O(n) zur Lösung dieses Problems minimal oder gibt es einen Algorithmus einer Komplexitätsklasse unterhalb O(n)? Begründen Sie Ihre Antwort. (5 Punkte)

3. Rekursion

Eine natürliche Zahl n in Dezimaldarstellung lässt sich auf folgende Weise rekursiv in ihre (hier mittels Dezimalzahlen simulierte) Binärdarstellung überführen:

$$\text{Binär}(n) := \begin{cases} n \bmod 2, & \text{falls } n <= 1 \\ 1 + \text{Binär}(n/2) * 10, & \text{falls } n>1 \text{ und } n \bmod 2 = 1 \\ \text{Binär}(n/2) * 10, & \text{sonst} \end{cases}$$

(./2 ist hier die ganzzahlige Division durch 2, .mod. die Modulo-Funktion (in C %), .*. die Multiplikation)

Gemäß dieser Definition kann eine rekursive C-Funktion (Ergebnistyp unsigned int) programmiert werden, die einen Parameter n vom Typ unsigned int besitzt und die Binärdarstellung von n per return als Ergebnis vom Typ unsigned int zurückgibt. (Die Division in C erfolgt bei ganzen Zahlen ganzzahlig)

Geben Sie den C-Code dieser Funktion an. (8 Punkte)

4. Datentyp Liste

Im Folgenden soll eine C-Funktion (Ergebnistyp list*) erstellt werden, die alle Vorkommen **bis auf das in der Zugriffsreihenfolge erste** einer bestimmten int Zahl n in einer Liste l von int Zahlen entfernt.
Ist die Liste leer, so ist auch das Ergebnis eine leere Liste.

Ist beispielsweise l = 6-3-5-7-7-2-7-1 (der Anfang der Liste sei hier rechts bei der 1) und hat n den Wert 7, so ist das Ergebnis 6-3-5-2-7-1.

Die zu bearbeitende Liste soll als Parameter vom Typ Pointer auf Liste an die Funktion übergeben werden. Die Zahl n soll ebenso ein Parameter der Funktion sein. Die zu bearbeitende Liste darf zerstört werden, die Reihenfolge der Elemente der Liste darf verändert werden.

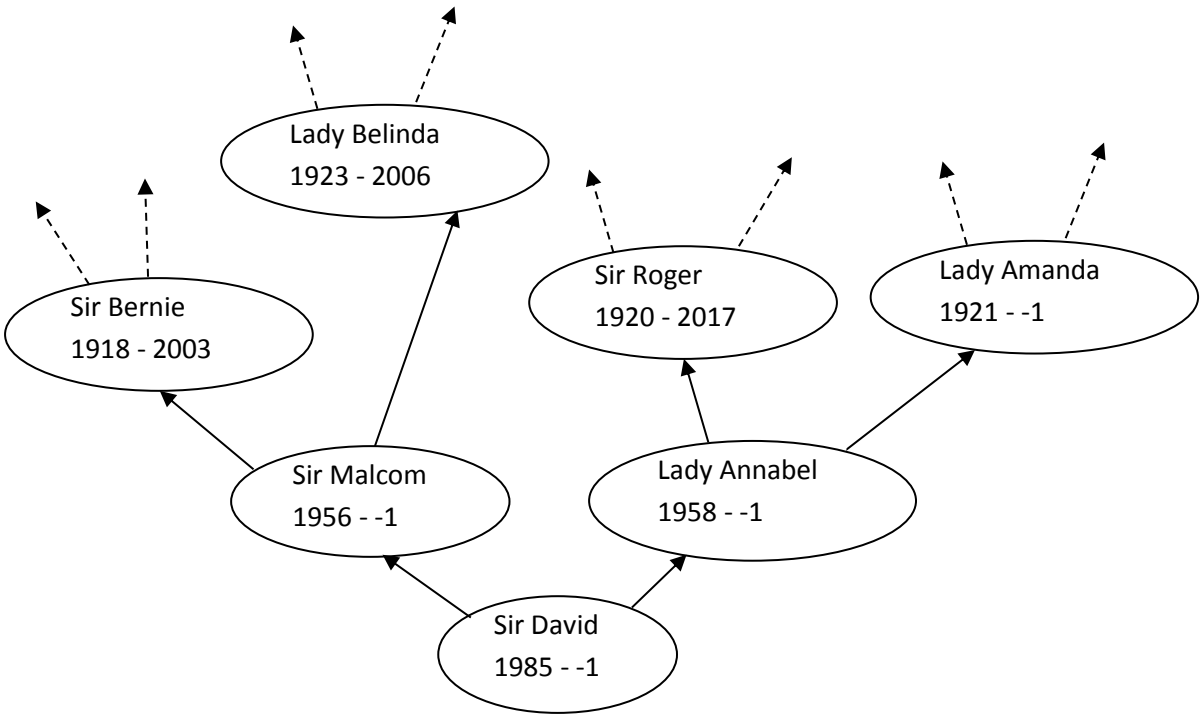
Zur Umsetzung der Anforderungen und Manipulation der Liste dürfen in der Funktion nur die, aus der Vorlesung und Übung bekannten, Listenfunktionen emptylist, isempty, tail, append und head (hier angepasst auf den Datentyp int) verwendet werden. Die Schnittstellen dieser Listenfunktionen sind im Anhang dargestellt und erläutert. Die zu bearbeitende Liste darf beim Bearbeiten zerstört werden.
Die Aufgabe kann iterativ (mittels einer oder mehrerer Schleifen) oder auch rekursiv gelöst werden.

Hinweis zur rekursiven Lösung:
Im Fall „rekursiv“ darf die Funktion einen weiteren Parameter haben, der angibt, ob das erste Auftreten der gesuchten Zahl bereits bearbeitet wurde.
(15 Punkte)

5. Datentyp Baum

Mittels binärer Bäume lassen sich Stammbäume darstellen. Dies nutzt der 27. Lord von Nottingham, Sir David Lindsay, aus, um in seinem eigenen Stammbaum, der viele Generationen bis in die Zeit um 500 nach Christus zurückreicht, computergestützt Ahnenforschung zu betreiben.

Im Bild unten ist ein Ausschnitt aus dem Stammbaum von Sir David dargestellt.



Um seine Forschungen mittels eines C Programms am Computer durchführen zu können hat Sir David folgende Definition für Stammbäume in der Programmiersprache C entwickelt:

```
typedef struct stammbaum
{ char* name;
  int geburtsjahr;
  int sterbejahr;
  stammbaum* vater;
  stammbaum* mutter;
};
```

Dabei wird in einem Element eines Stammbaums eine Person mit Name, Geburts- und Sterbejahr gespeichert, zwei Pointer auf Stammbaum zeigen auf den Stammbaum des Vaters der Person und auf den Stammbaum der Mutter der Person. Bei Personen, bei denen Vater oder Mutter nicht bekannt sind, hat der entsprechende Pointer den Wert NULL. Für Personen, deren Geburts- und/oder Sterbejahr nicht bekannt sind, wird für das Geburts- und/oder Sterbejahr der Wert 0 gespeichert. Für Personen, die noch leben, wird für das Sterbejahr der Wert -1

verwendet. Personen, die vor Christi Geburt lebten, werden hier nicht erfasst.

a.) Als erstes benötigt Sir David eine Funktion, die feststellt, ob eine Person einen Jahrhundertwechsel erlebt hat, also z.B. im 18. Jahrhundert geboren wurde und im 19. Jahrhundert gestorben ist. Die Funktion soll einen Pointer auf die Struktur stammbaum als Argument besitzen und den ganzzahligen (int) Wert 1 zurückgeben, falls die Person sicher einen Jahrhundertwechsel erlebt hat, andernfalls den (int) Wert 0. Das Ergebnis kann nach folgenden Überlegungen berechnet werden:
   - Ist das Geburts- oder Sterbejahr unbekannt (d.h. die entsprechende Komponente hat den Wert 0), so wird 0 als Ergebnis zurückgegeben.
   - Ist Sterbejahr geteilt durch 100 minus Geburtsjahr geteilt durch 100 ungleich 0 so ist das Ergebnis 1.
   - Falls eine Person noch lebt (Wert für Sterbejahr -1), wird das Jahr 2018 anstatt des Sterbejahrs verwendet.
     (10 Punkte)

b.) Basierend auf der Funktion aus Aufgabenteil a.) interessiert Sir David die Anzahl seiner bekannten Familienmitglieder inklusive seiner selbst, die sicher einen Jahrhundertwechsel erlebt haben.
Geben Sie eine C-Funktion an, die die Anzahl der Personen im Stammbaum mit der gesuchten Eigenschaft aus einem Stammbaum berechnet. Personen, bei denen das Geburts- oder Sterbejahr unbekannt ist, werden nicht berücksichtigt.
Der Stammbaum soll per Pointer auf Stammbaum an die Funktion übergeben werden.
Das Ergebnis soll als Wert vom Typ int per return zurückgegeben werden.
(16 Punkte)

Lösen Sie das Problem rekursiv und gehen Sie in folgenden Schritten vor:

- Wenn der Stammbaum leer ist (Parameter == NULL), so ist das Ergebnis, das Sie zurückgeben 0 (Terminierungsfall).
- Wenn der Stammbaum nicht leer ist, analysieren Sie die Wurzel, suchen im linken und rechten „Teil“-Stammbaum und verknüpfen die Teil-Ergebnisse geeignet.
  Das Gesamtergebnis geben Sie per return zurück.
- Eine wie auch immer geartete Fehlerbehandlung ist nicht erforderlich.

c.) Als nächstes interessiert Sir David, wie viele Generationen sein Stammbaum maximal zurückreicht. Geben Sie an, aus welcher Kenngröße eines (hier binären) Baums sich die Anzahl der Generationen im Stammbaum herleiten lässt. (4 Punkte)

6. Hashfunktionen

a. Geben Sie einen Vorteil eines Hashverfahrens im Gegensatz zu Sortierung an. (3 Punkte)

b. Geben Sie einen Nachteil eines geschlossenen Hashverfahrens an (2 Punkte)

c. Geben Sie eine Eigenschaft einer guten Hashfunktion an (2 Punkte)

d. Stellen Sie fest, ob die in Aufgabe 6c angegebene Eigenschaft in der unten angegebenen Hashfunktion, die einen Wert n auf 200 Behälter abbilden soll, vorhanden ist. Die Behälter sind mit 1 bis 200 durchnummeriert.
Begründen Sie Ihre Feststellung. (6 Punkte)

```
unsigned int f(unsigned int n)
{
  return(n%200) + 1;
}
```

Anhang

Der Datentyp Liste von int hat folgende Signatur:

1. list* emptylist()
   Erzeugt eine leere Liste und gibt einen Pointer darauf zurück
2. int isempty (list *l)
   Gibt 1 zurück, falls l leer ist, sonst 0.
   Die Liste l bleibt unverändert */
3. list* tail (list *l)
   Löscht das erstes Element aus l, Voraussetzung: Liste ist nicht leer.
4. list* append (list *l, int e)
   Fügt das Element e vorne an l an.
5. int head (list * l)
   Liefert erstes Element der Liste l, l bleibt unverändert,
   Voraussetzung: Liste l ist nicht leer.