AP-Klausur vom 06.10.2001
Lösung

Aufgabe 1

a.) Betrachten Sie den folgenden Ausschnitt eines Programms, das Vokale zählt:

```
char z
int az = 0, ez = 0, iz = 0, oz = 0, uz = 0;
switch (z)  {
      case 'A' : ++az;
      case 'a'  : ++az; break;
      case 'E' :
      case 'e' : ++ez; break;
      case 'I'  :
      case 'i'  : ++iz;
      case 'O' :
      case 'o' : ++oz; break;
      case 'U' :
      case 'u' : ++uz;
      }
```

Bitte kreuzen Sie die richtigen Antworten an:

[X] Der Programmausschnitt ist syntaktisch Korrekt
[ ] 'A' wird korrekt gezählt
[ ] Das zählen der Vokale funktioniert nicht, denn alle Zähler haben immer den gleichen Wert
[X] 'E' wird korrekt gezählt
[X] Wenn ein 'I' gelesen wird, dann wird der Zähler für 'o' um eins erhöht

b.) Betrachten Sie die folgende if - Anweisung

```
if ( n > 0 )
      if ( i != 0 ) j = 1;
else result = 1;
```

Bitte kreuzen Sie die richtigen Antworten an:

[ ] Der Programmausschnitt ist syntaktisch nicht ganz korrekt, denn es fehlt ein else - Zweig
[ ] Der Programmcode ist gut und übersichtlich formatiert und daher leicht lesbar
[X] Wenn n > 0 und i = 0 gilt, dann wird result = 1 ausgeführt
[ ] Wenn n <= 0 gilt, dann wird result = 1 ausgeführt
[ ] Wenn n = = 0 und i = = 0 gilt, dann wird j = 1 ausgeführt

c.) Formulieren Sie folgendes als Bedingung einer if - Anweisung:
"Ein Jahr ist ein Schaltjahr, wenn die Jahreszahl durch 4 aber nicht durch 100 teilbar ist.
Wobei noch die Ausnahme gilt, dass Jahre, die durch 400 teilbar sind, Schaltjahre sind."

Bitte schreiben Sie hier die if - Anweisung hin, die ausgibt,
ob es sich um ein Schaltjahr handelt oder nicht:

```
int jahr;

if  (((jahr % 4 == 0)&&(jahr % 100 != 0)) || (jahr % 400 == 0))
{
      printf("%I ist ein Schaltjahr", jahr);
}
else
{
      printf("%I ist kein Schaltjahr", jahr);
}
```

d.) Betrachten Sie die folgende Funktion

```
int xyz (char s [ ]) {
      int i = 0;
      while (s [ i++ ]);
      return i;
      }
```

Bitte kreuzen Sie die richtigen Antworten an:

[x] Der Programmausschnitt ist syntaktisch korrekt
[ ] Die while - Schleife wird nie durchlaufen. Die Funktion gibt also immer 0 zurück
[x] Die while - Schleife testet, ob das terminierende Nullbyte erreicht ist
[ ] Die Funktion berechnet die Stringlänge
[x] Die Funktion berechnet die Anzahl der Zeichen im Array (incl. des Nullbytes)

Aufgabe 2

a.) Es sei "feld" ein Array mit "s" Integerzahlen.
Schreiben Sie eine rekursive Funktion "sum", die die Summe der Zahlen als Rückgabewert liefert.
Als Eingabeparameter erhält die Funktion "feld" und "s".

Lösung:

```
int sum(int *feld, int s) {
  /* wenn das Restfeld nur noch eins lang ist,
     dann bleibt nur die erste Zahl -> feld[0] */
  if (s == 1) return feld[0];
  else
  /* es sind noch mehrere Zahlen da, also
     letzte Zahl plus Summe des Restfeldes */
    return (feld[s-1] + sum(feld, s-1));
}
```

b.) Das Array "feld" sei mit den Werten 1,2 und 3 initialisiert:

int feld [ ] = { 1,2,3 };

Schreiben Sie bitte die rekursive Aufruffolge der Funktion "sum (feld,3)" hier hin:

Lösung:

```
int summe   =     sum(feld, 3)
            =     feld[2] + sum(feld, 2)
            =     feld[2] + feld[1] + sum(feld, 1)
            =     feld[2] + feld[1] + feld[0]
            // keine weiteren rekursiven Aufrufe, weil die
            // Abbruchbedingung erreicht ist.
            = 3 + 2 + 1
            = 6
```

c.) Bitte ergänzen Sie die folgenden leeren Textstellen

int a[ ] = {1,2,3,4}; Die Array-Elemente werden mit 1, 2, 3, 4 initialisiert
Es wird ein Array der Größe 4 angelegt. (Indizes 0..3!)

int a[3] = {1,2,3,4}; ergibt eine Fehlermeldung, weil einem Array der Größe 3 vier Elemente zugewiesen werden -> Bereichsüberschreitung

char z[ ] = {'A','B','C'}; Die Array-Elemente werden mit A, B, C initialisiert
Es wird ein Array der Größe 3 angelegt. (Indizes sind 0..2)

char z[ ] = "ABC"; Die Array-Elemente werden mit A, B, C, \0 initialisiert
Es wird ein Array der Größe 4 angelegt.

char z[6] = "ABCDEF"; ergibt eine Fehlermeldung, weil einem Array der Größe 6 sieben Elemente zugewiesen werden (6 Zeichen und ein Nullbyte!)

d.) Schreiben Sie eine Funktion "loesche(s,c)", die alle Vorkommen des Zeichens "c" im String "s" löscht.
Der Resultatstring wird im Parameter "s" zurückgegeben.

```
void loesche(char *s, char c) {
  int i = 0, j = 0;
  /* solange das Ende des Strings nicht erreicht wird */
  while (s[i]) {
    /* aktuelles Zeichen = gesuchtes Zeichen */
    if (s[i] == c) {
       /* ja, dann schiebe alle Nachfolgenden eins vor */
       for (j = i; s[j+1]; j++)
           s[j] = s[j+1];
       s[j] = '\0';
       /* ist immernoch das aktuelle = gesuchtem, dann darf man
          i nicht erhöhen sondern muss direkt in den nächsten
          Schleifendurchlauf gehen -> continue*/
       if (s[i] == c) continue;
    }
    i++;
  }
}
```

Aufgabe 3 -= Objektorientierung =--= Ehses=- -= nur Java beispiele abgeschrieben =-

a.) Welche Deklarationen, die innerhalb einer Klasse (in Java: abstract class) stehen, sind richtig bzw. falsch (... steht natürlich für richtige Anweisungen)

	richtig	falsch
final void p1() { ... }	[x]	[ ]
final void p2();	[ ]	[x]
abstract void p3();	[x]	[ ]
abstract final void abc();	[ ]	[x]
abstract void p5() { ... }	[ ]	[x]

b.) Gegeben sind die Abstrakte (Basis-)Klasse "Abstract" und die davon abgeleitete konkrete Klasse "Derived".
Der Konstruktor der beiden Klassen hat jeweils einen int - Parameter.
Welche der folgenden Anweisungen sind richtig / falsch?

	richtig	falsch
Abstract [ ] a1 = new Abstract [10];	[x]	[ ]
Abstract a2 = new Abstract (5);	[ ]	[x]
Derived [ ] a3 = new Derived [10];	[x]	[ ]
Abstract [ ] a4 = new Derived [10];	[x]	[ ]
Abstract a5 = new Derived (5);	[x]	[ ]
Abstract [ ] a6 = new Derived [10](5);	[ ]	[x]
a1 [2] = new Abstract (5);	[ ]	[x]
a1 [3] = new Derived (5);	[x]	[ ]

c.) Welche der folgenden Aussagen sind richtig bzw. falsch?

	richtig	falsch
final class Abc bedeutet, dass die Klasse Abc nicht verändert werden darf	[x]	[ ]
abstract class Abc bedeutet, dass die Klasse Abc nicht verändert werden darf	[ ]	[x]
Für ein interface Abc, darf man Variable (Abc x;)anlegen	[x]	[ ]
Eine Java - Klasse darf mehrere Interfaces implementieren	[x]	[ ]
Ein static - Attribut darf nicht verändert werden	[ ]	[x]
Späte Bindung untersucht zur Laufzeit, zu welcher Klasse ein Objekt gehört	[x]	[ ]

Aufgabe 4 -= Algorithmen =-

Gegeben ist die folgende Klasse für einen Knoten eines Binärbaums.
Bei Aufgabe a.) und b.) werden nur Lösungen gewertet, die exakt der Deklaration entsprechen.

```
class Node {
      Node left;
      Node right;
      double value;

      int count() { ... }
      String toString() { ... }
      static void max(Node root) { ... }
}
```

a.) [Baum] Schreiben Sie die Methode count(), die die Anzahl aller Knoten im Baum ermittelt.

Lösung:

```
int count() {
  int l = 0, r = 0;
  // Knoten links zaehlen
  l = (left == null) ? 0 : left.count();
  // Knoten rechts zaehlen
  r = (right == null) ? 0 : right.count();
  // Summe ist aktueller (1) plus links plus rechts
  return (1 + l + r);
}
```

b.) Schreiben Sie eine Methode toString(), die für alle Knotenwerte (value) eines Baumes in Preorderreihenfolge durch Leerzeichen getrennt zu einem String zusammenfügt und diesen zurückgibt (Ausgabe, z.B. System.out.println(root.toString()).

```
String toString() {
  // linken und rechten Teilbaum einzeln zu Strings machen:
  String l = (left == null) ? "" : left.toString();
  String r = (right == null) ? "" : right.toString();
  // bei preorder wird der aktuelle (eigene) Wert zuerst augegeben:
  return (value + l + r);
  // value wird dabei durch den Konkatenationsoperator zu String konvertiert
}
```

c.) [Queue] Angenommen, Sie haben bereits eine Klasse List, die eine verkettete Liste implementiert, mit einem Standardkonstruktor und den Methoden
void add(int value) (fügt value in die Liste ein),
int getFirst() (gibt value Inhalt des 1. Listenelements zurück) und
void removeFirst() (entfernt das 1. Element aus der Liste).
Implementieren Sie komplett eine Klasse IntQueue, die einen Konstruktor und eine Methode zum Einfügen (put) und eine Methode zum Herausnehmen (get) eines Wertes enthält.
Die Klasse IntQueue soll sich wie eine in der Vorlesung besprochene Warteschlange (für int als Inhalt) verhalten (FIFO).

```
public class IntQueue {
  private List liste;

  // Konstruktor
  public IntQueue() {
    // lege eine neue Liste an
    liste = new List();
  }

  // fuegt eine Zahl in die Queue ein
  public void put(int n) {
    // davon ausgehend, dass add(int) am Ende einfuegt
    l.add(n);
  }

  // holt das nächste Element aus der Queue
  public int get() {
    int temp = liste.getFirst();
    liste.removeFirst();
    return temp;
  }
}
```

d.) [Hashtabelle] Beschreiben Sie die Idee der Hashtabelle und ein mögliches Verfahren zur Behandlung von Kollisionen.
Wie wächst die Laufzeit, wenn die Tabelle größer wird?

Mögliche Lösung:
Bei der Hashtable handelt es sich um eine Art und Weise, Daten in einem Array im Speicher abzulegen, sodass Zugriffe darauf möglichst schnell sind. Dabei wird für die abgelegten Objekte / Daten ein Hashwert (durch eine entsprechende Funktion) berechnet, der dann sozusagen den Ort angibt, wo das Objekt abgelegt wird; also ein Index für ein Array. Dadurch ist die Dauer, die benötigt wird, um an einem Eintrag dran zu kommen, unabhängig von der Größe der Tabelle selbst, also O(1).
Kommt eine Kollision vor, weil für zwei verschiedene Objekte derselbe Hashwert errechnet wurde, gibt es die Möglichkeit, linear zu verschieben, d. h. den nächsten freien Platz im Array zu belegen. Eine andere Möglichkeit ist es, sogenannte „buckets“ anzulegen, bspw. In Form von linearen Listen, die an den jeweiligen Indizes „hängen“. Treten in einem derartigen Fall jedoch sehr viele Kollisionen auf – z. B. aufgrund eines hohen Füllungsgrades der Tabelle – leidet die Zugriffszeit darunter, weil in den buckets nur linear gesucht wird, also mir O(n).
Die Hashtable aus dem package java.util vergrößert sich selbständig, wenn sie einen gewissen Füllstand erreicht hat.

e.) [Liste] Schreiben Sie die Deklaration für eine einfach verkettete Liste auf (Klasse List und Klasse Node). Schreiben Sie den Konstruktor und eine Methode zum Einfügen eines Wertes in die Liste (ob am Anfang oder am Ende ist Ihre Sache) vollständig hin.
Alle anderen Methoden sollen Sie weglassen.

```
public class List {
  // Start der Liste - root
  private Node root;

  // innere Klasse, einzelner Listenknoten
  class Node {
    Object content;
    Node next;

    // Konstruktor der nested class
    Node(Object c, Node n) {
      content = c;
      next = n;
    }
  }

  // Konstruktor der Liste
  public List() {
    // Liste zu Beginn leer
    root = null;
  }

  // fuegt ein Object in die Liste ein (am Anfang)
  public void put(Object obj) {
    if (root == null) root = new Node(obj, null);
    else {
      // am Anfang der Liste einfuegen mit root als Nachfolger
      Node temp = new Node(obj, root);
      // neuen Listenanfang festlegen
      Root = temp;
    }
  }
  // hier weitere Methoden
}
```