Probeklausur Informatik I

Erklärungen

- bei der E-Klausur werden die Aufgaben ILIAS-typisch auf einzelnen Seiten angezeigt (gut lesbar)
- außerdem gibt es bei der E-Klausur diese Möglichkeiten:
  - Vergrößerung des Texts mit Strg-Mausrad
  - Eingabefenster für Freitexte an Ecke rechts-unten aufziehbar
  - Antwortoptionen (nachfolgend in blauer Schrift) werden in Drop-Down-Menüs angezeigt
- bei der Klausur wird ein leeres Blatt für Notizen bereitgelegt, Notizen werden nicht bewertet
- im hier vorliegenden Layout sind Texte mit sehr langen Zeilen abgeschnitten, hier die Ergänzungen (fehlender/unlesbarer Text kursiv)

  Aufgabe 2.2 ... leistet: Es *wird die Summe der ganzen* Zahlen ...

  Aufgabe 12 ... zutreffenden A *ntworten aus (jeweils 1.5P für* die ...

  Aufgabe 13.1 ... weitere as *ymptotische Analyse* geeignet?

  Aufgabe 13.2 ... sehen *typischerweise so aus:*

  Aufgabe 14 ... vorkomm *t*

  ... Fol *gendes leisten:*

  ... Bereichsbedingung erf *üllt ist, dann mit zweiter Schleife* in einem Hilfsfeld ...

  ... Feld anl *egen und in Schleife die* Komponenten ...
- zu Aufgaben 14/15 gehören Freitextfelder, die hier aber nicht zu sehen sind
- Punkte

Aufgabe	erreichbar
1	4
2	5
3	6
4	5
5	6
6	4
7	6
8	6
9	6
10	6
11	2
12	3
13	6
14	12
15	13
Summe	90

Liste der Antworten

1. Allgemeine Grundkenntnisse (I1-ID: rjyb60x0u5h0)

1. Geben Sie die richtige Antwortkombination an (1P für die richtige/0P für jede falsche Kombination):
   - (A) Spezifikationen bestehen aus einer präzisen Beschreibung der erlaubten Eingaben.
   - (B) Algorithmen sind endliche Vorschriften, die Rechenabläufe schrittweise beschreiben.
   - (C) Compiler übersetzen Quellcode erst in Assembler, dann in Unix-Code.
   - (D) Das Java-Programm ist auf dem System lauffähig, wenn darauf der entsprechende Compiler installiert ist.

   z.B. A und B und C und D, ..., nicht A und nicht B und nicht C und nicht D
2. Geben Sie die richtige Antwortkombination an (1P für die richtige/0P für jede falsche Kombination):
   - (A) Bei der Entwicklung eines Programms muss nach jedem Compilieren editiert werden.
   - (B) Interaktive Programme können Daten von der Standard-Eingabe erhalten.
   - (C) Am CMD-Flag ist erkennbar, ob die Speicherzelle eine Zahl oder einen Befehl enthält.
   - (D) Eine *Pipe* funktioniert im wesentlichen wie eine Queue.

   z.B. A und B und C und D, ..., nicht A und nicht B und nicht C und nicht D
3. Geben Sie die richtige Antwortkombination an (1P für die richtige/0P für jede falsche Kombination):
   - (A) Durch Eingabeumleitung kann man gespeicherte Daten an ein interaktives Kommando leiten.
   - (B) Informationsverarbeitung zerfällt in die Schritte Codierung, Datenverarbeitung und Decodierung.
   - (C) Mit 4 Hexadezimalziffern kann man mehr als 130 000 Werte darstellen.
   - (D) Der Bruch 4/5=0.8 kann exakt als endlicher Dualbruch dargestellt werden.

   z.B. A und B und C und D, ..., nicht A und nicht B und nicht C und nicht D
4. Geben Sie die richtige Antwortkombination an (1P für die richtige/0P für jede falsche Kombination):
   - (A) Ganzzahlarithmetik ist normalerweise schneller als Gleitkomma-Arithmetik.
   - (B) Strukturierte Programmierung ist die Programmierung mit strukturierten Daten.
   - (C) Rekursion benötigt mehr Speicher auf dem Stack als Iteration.
   - (D) Die JVM interpretiert jeden Bytecodebefehl und prüft dessen Syntax.

   z.B. A und B und C und D, ..., nicht A und nicht B und nicht C und nicht D

2. Einige Java-Grundkenntnisse (I1-ID: u1j7g25165h0)

1. Ordnen Sie Begriffe zeitlich (0.25P für jede korrekte Position)

   Algorithmus - Compiler - Editor - Lösungsidee - Quelltext - Spezifikation - Testen - Zielcode

2. Füllen Sie die Lücken, so dass das Programm mit zwei beliebigen Zahlwerten A und B als Kommandozeilenparametern aufgerufen Folgendes leistet: Es Zahlen ausgeben, die innerhalb des *abgeschlossenen* Intervalls [A,B] liegen. (0.25P für jede korrekte Angabe)

```
public class Intervallsumme {
    public static void main(String[] args) {
            A          ,
            min = B                  (args[0]),
            max = C                  (args[1]);
        int D       ;
        while ( E         ) {
             F       ;
             G     ;
        }
        H
    }
}
```

   <Lücke> (<Anzahl der Möglichkeiten>): <Mögl.1>, <Mögl.2>, ...
   A(4): double s=A, int s=A, double s=0, int s=0
   B(2): StdIn.readInt, Integer.parseInt
   C(2): StdIn.readInt, Integer.parseInt
   D(4): i=min, n=min, s=min, t=min
   E(4): min<=n, min<=s, n<=max, s<=max
   F(4): n+=s, s+=n, n=n+1, s = s+1
   G(4): n++, s++, n+=s, s+=n
   H(4): return n, return s, System.out.println(n), System.out.println(s)

3. Wieviele Bytes beansprucht eine Variable?
   - `int`: A
   - `char`: B
   - `float`: C
   - `boolean`: D

   Für jede Lücke diese 6 Möglichkeiten:
   1, 2, 4, 8, 16, nichts von diesen

   (0.25P für jede richtige Antwort)

3. Grundlegende Datentypen (I1-ID: ksu13v0075h0)

1. Geben Sie die richtige Antwortkombination an (1P für die richtige/0P für jede falsche Kombination):
   - (A) Das Leerzeichen ist ein druckbares Zeichen.
   - (B) iso8859-1 ist eine Teilmenge von ASCII.
   - (C) `'C'++` liefert das gleiche Ergebnis wie `'C'+1`.
   - (D) `"C" + 1` ist *kein* erlaubter Java-Ausdruck.
2. Geben Sie die richtige Antwortkombination an (1P für die richtige/0P für jede falsche Kombination):
   - (A) `int` ist ein Ganzzahldatentyp, `char` nicht.
   - (B) `int x = Integer.MAX_VALUE + 1;` ist eine erlaubte Anweisung.
   - (C) Der Wert von `(( 3 < 7 ) && ((17 + 4) % (12 - 5)) != 0)` ist `false`.
   - (D) Der Wert von `( 3 + 1/3 == 3 - 1/3)` ist `false`.
3. Geben Sie die zutreffenden Antworten an:

<Lücke> (<Anzahl der Möglichkeiten>): <Mögl.1>, <Mögl.2>, ...
A(4): 5, 6, keine (wegen Syntaxfehler), keine (wegen logischem Fehler)

- (A) `String s = "Hallo", System.out.println(s.length);`
  Ausgabe?
- (B) `System.out.println(10e-1f);`
  Ausgabe?

B(4): 1, 1.0, keine (wegen Syntaxfehler), keine (wegen logischem Fehler)

- (C) `2 * (short) 0xCafe`
  Typ des Ausdrucks?
- (D) `double y = Double.MIN_VALUE; System.out.println(y/2);`
  Ausgabe?

C(4): short, int, undefiniert (wegen Syntaxfehler),
undefiniert (wegen logischem Fehler)

D(4): 0, 0.0, keine (wegen Syntaxfehler), keine (wegen logischem Fehler)

- (E) `int x = Integer.MAX_VALUE, y = Integer.MAX_VALUE++, z = x+y;`
  Wert von `z`?

E(4): -1, +1, undefiniert (wegen Syntaxfehler),
undefiniert (wegen logischem Fehler)

- (F) `System.out.println( (double) (1/2));`
  Ausgabe?

F(4): 0.0, 0.5, keine (wegen Syntaxfehler), keine (wegen logischem Fehler)

4. Geben Sie die richtige Antwortkombination an (1P für die richtige/0P für jede falsche Kombination):
   - (A) `int[] z = {1}; System.out.println(z);` produziert die Ausgabe 1.
   - (B) `length` ist ein `final`-Attribut von Feldern.
   - (C) Durch `int[] a = {1, 2, 3}, b = new int[];` werden zwei neue Felder angelegt.
   - (D) Durch `int[][][][] x = new char[32][32][32][16];` wird etwa 1Mb Heap-Speicher reserviert.

4. Integer-Arithmetik (I1-ID: 5wa3il81gdg0)

Sei `d` eine `int`-Variable mit dem Wert `Integer.MAX_VALUE` (also 2147483647). Welche Textausgabe erzeugen diese Anweisungen?:

```
1:  System.out.println(-d);     //
2:  System.out.println(d-1);    //
3:  System.out.println(1+2*d);  //
4:  System.out.println(-(2*d)); //
```

Achten Sie darauf, dass Ihre Antworten NUR Dezimalziffern enthalten (sowie bei Bedarf Vorzeichen)!

5. Iteration in Feldern (I1-ID: t75bm8e005h0)

```
int[] p = { 1, 4, 0, 3, 6, 2, 7, 5 };
int[] c = { 0, 0, 0, 0, 0, 0, 0, 0 };
int[] q = { 0, 0, 0, 0, 0, 0, 0, 0 };
for (int i = 0; i < 8; i++)
    c[p[i]] = c[p[i]] + 1;
for (int i = 0; i < 8; i++)
    q[p[i]] = i;
```

Tragen Sie die Werte der Feldkomponenten *nach* Ausführung des Java-Fragments ein:

Index	0	1	2	3	4	5	6	7
c								
q								

6. Felder: Fehler oder...? (I1-ID: o2vav4a1gdg0)

Erzeugt untenstehende Klasse

- einen Compilezeitfehler? Falls ja, so tragen Sie die Nummer der fehlerhaften Zeile ein, ansonsten einen Bindestrich (-):
- einen Laufzeitfehler? Falls ja, so tragen Sie die Nummer der fehlerhaften Zeile ein, ansonsten einen Bindestrich (-):
- eine Ausgabe? Falls ja, so tragen Sie Ausgabe ein, ansonsten einen Bindestrich (-):

```
1: class FehlerOderKeiner {
2:     public static void main(String[] args) {
3:         int[] a = {2,3,5,7,11}; int[] b = { a[2], a[2]};
4:         b = a;
5:         b[1] = b[0];
6:         System.out.println(a[0] != a[1]);
7:     }
8: }
```

7. Rekursion (I1-ID: qft7dic1y4h0)

Betrachten Sie folgende Klasse:

```
 1: class Myst {
 2:     static void ery(int n) {
 3:         if (n == 0)
 4:             return;
 5:         ery( n-1);
 6:         A();
 7:         yre( n-1);
 8:     }
 9:     static void yre(int n) {
10:         if (n == 0)
11:             return;
12:         ery( n-1);
13:         B();
14:         yre( n-1);
15:     }
16:     static void A() {
17:         System.out.print("A");
18:     }
19:     static void B() {
20:         System.out.print("B");
21:     }
```

Fragen auf nächster Seite
<Lücke> (<Anzahl der Möglichkeiten>): <Mögl.1>, <Mögl.2>, ...
1. (10): AABAABAA,
   BBABBAAB,
   ABAAABBA,
   BABBBAAB,
   AABAABBA,
   BBABBAAB,
   AABBAABA,
   BBAABBAB,
   AABABABB,
   BBABABAA
2. (12): 15, 16, 17,
   49, 50, 51,
   127, 128, 129,
   255, 256, 257
3. (9): Zeile 3, Zeile 4, Zeile 5, Zeile 6, Zeile 7,
   Zeile 10, Zeile 11, Zeile 12, Zeile 13
4. (7): -3, -2, -1, 0, 1, 2, 3

```
22:     public static void main(String[] args) {
23:         int n = Integer.parseInt(args[0]);
24:         ery(n);
25:         System.out.println();
26:     }
27: }
```

1. Geben Sie die ersten 8 Zeichen der Ausgabe zum Aufruf `java Myst 4` an:

2. Aus wievielen Zeichen besteht der Ausgabestring beim Aufruf `java Myst 7` (ohne den abschließenden Zeilenwechsel durch Zeile 25)?

3. Wird Zeile 14 durch `yre( n-2);` ersetzt, so führt beispielsweise der Aufruf `java Myst 4` zu einem Fehler.
   Es genügt, ein einzelnes Zeichen in einer *anderen* Zeile zu ändern, um diesen Fehler nicht mehr hervorzurufen. Um welche Zeile handelt es sich?

4. Werde Zeile 12 durch `ery( n-2);` und zusätzlich Zeile 14 durch `yre( n-2)` ersetzt.
   Geben Sie das maximale Argument `n` an, bei dem der Aufruf `java Myst n` *nicht* zu einem Fehler führt:

8. Stack und Queue in Aktion (I1-ID: 8303049035h0)

1. Auf einem Stack werden *in gemischter Reihenfolge* 10 Push- und 10 Pop-Operationen ausgeführt. Es ist bekannt, dass die Push-Operationen die Zahler Reihenfolge einspeichern und dass bei jedem Pop der ausgespeicherte Wert ausgegeben wird. Welche Ausgabelisten sind möglich?

   Wählen Sie aus den beiden Blöcken jeweils die zutreffende Antwort aus (jeweils 2P für die richtige Kombination/0P für jede andere).

   - (A) 1 2 3 4 5 6 9 8 7 0
   - (B) 0 4 6 5 3 8 1 7 2 9

   beide, nur A, nur B, keine

   - (C) 1 4 7 9 8 6 5 3 0 2
   - (D) 2 1 4 3 6 5 8 7 9 0

   beide, nur C, nur D, keine

2. Auf einer Queue werden *in gemischter Reihenfolge* 10 Enqueue- und 10 Dequeue-Operationen ausgeführt. Es ist bekannt, dass die Enqueue-Operation dieser Reihenfolge einspeichern und dass bei jedem Dequeue der ausgespeicherte Wert ausgegeben wird. Welche Ausgabelisten sind möglich?

   Wählen Sie aus den beiden Blöcken jeweils die zutreffende Antwort aus (jeweils 1P für die richtige Kombination/0P für jede andere).

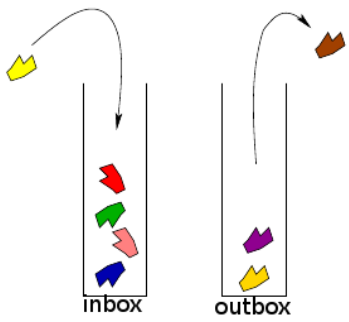
   - (E) 4 6 8 7 5 3 2 9 0 1
   - (F) 2 5 6 7 4 8 9 3 1 0

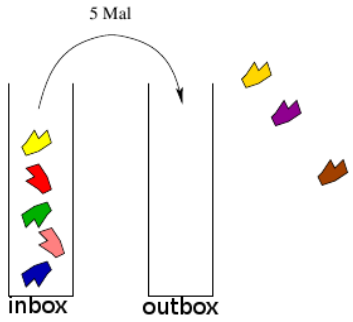
   beide, nur E, nur F, keine

   - (G) 0 1 2 3 4 5 6 7 8 9
   - (H) 4 3 2 1 0 5 6 7 8 9

   beide, nur G, nur H, keine

9. Queue aus zwei Stacks bauen (I1-ID: a4g40y31x4h0)





Mit zwei Stacks kann man eine Queue implementieren:

- `enqueue(...)` entspricht Einspeichern in `inbox`
- `dequeue()` entspricht Ausspeichern aus `outbox`.

Unter Verwendung dieser API

`public class Stack`	
`boolean empty()`	testet, ob dieser Stack leer ist
`void push(String s)`	speichert `s` in diesem Stack
`String pop()`	liefert zuletzt gespeicherten String aus diesem Stack und entfernt ihn

sähe die entsprechende Klasse z.B. so aus:

```
public class ZSQueue { // Zwei-Stack-Queue
    private Stack inbox, outbox;
    public ZSQueue() { ... }
    public boolean empty() { ... }        // Leerheitstest
    public void enqueue(String s) { ... } // einspeichern
    public String dequeue() {             // ausspeichern und liefern
        // Rumpf (Ihre Aufgabe)
    }
}
```

zur Auswahl angebotene Codezeilen:

```
if (empty())
if (empty(outbox))
if (inbox.empty())
if (outbox.empty())
if (!empty())
if (!inbox.empty())
inbox.push( outbox.pop());
outbox.pop(inbox.push());
outbox.push(inbox.pop());
return inbox.pop();
return inbox.push();
return outbox.pop();
return;
System.out.println("Queue ist leer!);
throw new RuntimeException("Queue ist leer");
while (!inbox.empty())
while (outbox.empty())
while (inbox.empty())
```

(siehe nächste Seite)

Stellen Sie die Zeilen für den Rumpf der Methode `dequeue()` richtiger Reihenfolge zusammen:

- ist die Queue leer, ist Ausspeichern nicht möglich!
- teste ob `outbox` leer ist und fülle in diesem Fall den kompletten Inhalt von `inbox` um in `outbox`, dann ein Element aus `outbox` ausspeichern und liefern.

10. Vermischtes zu Klassentypen (I1-ID: t5k8w9h0frg0)

1. Objekte verstehen:

```
public class Spieler {
    private int score;
    public Spieler( int s) {
        this.score = s;
    }
}
```

Zwei `Spieler`-Objekte werden instanziiert:

```
Spieler a = new Spieler(50), b = new Spieler(50);
```

Welchen Wert hat nun der Ausdruck a == b?

2. Fremden Code deuten:

```
1:  public class FirstBpp extends hepno.users.Frame {
2:      private hepno.users.Wring wrags;
3:      public FirstBpp() {
4:          wrags = new hepno.users.Wring();
5:      }
6:      public static void main(String[] args) {
7:          FirstBpp app = new FirstBpp();
8:      }
9:  }
```

Was bewirkt Zeile 2?:

3. Statische Methoden und Instanzmethoden:

```
public class Klasse {
    private final int zahl = 10;
    double a() {
        System.out.println(zahl);
        return c();
    }
    void b() {
        System.out.println(this);
    }
    double c() {
        double r = Math.random();
        System.out.println(r);
        return r;
    }
    void d() {
        a();
        a();
    }
    int e() {
        return zahl;
    }
}
```

Welche der obigen Methoden ist ihrer Natur nach eher eine statische Methode (und sollte daher mit `static` versehen werden)?

4. Fehlersuche:

```
public class Test {
    private int testCount;
    public static int getCount(){
        return testCount;
    }
    public Test(){
        testCount++;
    }
}
```

Obiger Code verursacht einen Compilezeitfehler. Was ist das Problem?

5. Tracing:
Hinweis: Beachten Sie, dass man über Objektreferenzen auch auf Klassenvariablen zugreifen kann.

```
public class Tset {
    static int tsetCount = 0;
    public Tset(){
        tsetCount++;
    }
    public static void main(String [] args){
        Tset t1 = new Tset();
        System.out.print(t1.tsetCount + ", ");
        Tset t2 = new Tset();
        System.out.print(t1.tsetCount + " "+ t2.tsetCount + ", ");
```

<Lücke> (<Anzahl der Möglichkeiten>): <Mögl.1>, <Mögl.2>, ...
1. (5): true, false, das ist Compile-abhängig, das ist speicherabhängig,
   nichts von denen
2. (5): auf hepno.users.Wring kann mit wrags zugegriffen werden,
   wrags wird als Instanzvariable vereinbart,
   hepno.users.Wring wird als private-Objekt deklariert
   wrags instanziiert die Klasse hepno.users.Wring,
   nichts von denen
3. (5): a, b, c, d, e
4. (6): testCount wurde nicht initialisiert,
   auf testCount wurde nicht zugegriffen,
   getCount ist statisch,
   testCount ist private,
   Test() hat keinen Ergebnistyp,
   nichts von denen
5. (6 in einzelnen Zeilen angegeben):
   0, 0 0, 0 0 0
   1, 1 1, 1 1 1
   1, 2 2, 3 3 3
   1, 2 3, 4 5 6

   nichts von denen
6. (4 in einzelnen Sätzen angegeben):
In einer Klasse darf es Methoden geben, die den gleichen Namen haben wie der Konstruktur.
In einer Klasse darf es Methoden geben mit gleichem Namen und gleichen Ergebnistypen, aber unterschiedlich vielen Argumenten.
In einer Klasse darf es Methoden geben, die den gleichen Namen und gleiche Argumenttyplisten haben, aber unterschiedliche Ergebnistypen.
In einer Klasse darf es mehrere Konstruktoren mit dem gleichen Namen geben.

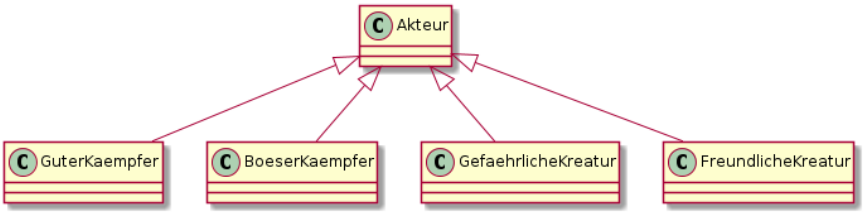
```
            Tset t3 = new Tset();
            System.out.print(t1.tsetCount+ " "+ t2.tsetCount+ " "+ t3.tsetCount);
            System.out.println();
        }
    }
```

Welche Ausgabe produziert der Code?

6. Begriffsverständnis: Wählen Sie aus dem Menü die einzige Aussage aus, die NICHT korrekt ist.

11. OOP-Entwurf (I1-ID: lf9g4np0qtg0)

In einer Spiele-Software gibt es folgende Hierarchie von Java-Klassen, die die verschiedenen Sorten von Akteuren modelliert:



Die (hier nicht gezeigte) Klasse `Spiel` benutzt die zugreifbaren Methoden obiger Klassen, um den Ablauf zu gestalten.

Wie ist (unter Erhalt der gezeigten Klassenbeziehungen) folgende Zusatzanforderung am besten zu modellieren?
Alle Kaempfer-Objekte aber keine Kreatur-Objekte sollen mit einer Methode benutzeWaffe() ausgestattet werden.

- Die Methode `benutzeWaffe()` nur in `GuterKaempfer` und in `BoeserKaempfer` wie benötigt implementieren.
- `benutzeWaffe()` mit leerem Rumpf in `Akteur` implementieren, in `GuterKaempfer` und `BoeserKaempfer` geeignet überschreiben.
- `benutzeWaffe()` in `Akteur` implementieren, in `GefaerlicheKreatur` und `FreundlicheKreatur` mit `private` kennzeichnen.
- Interface `Kaempfer` mit `benutzeWaffe()` definieren und durch `GuterKaempfer` und `BoeserKaempfer` implementieren lassen.
- Weitere Klasse `Kaempfer` schreiben, die `benutzeWaffe()` implementiert, `GuterKaempfer`, `BoeserKaempfer` davon erben lassen.

12. OOP-Verständnisfragen (I1-ID: fnkdcz9075h0)

Sei p eine Referenzvariable vom Typ X. Was kann alles der dynamische Typ von p sein? Wählen Sie aus den beiden Blöcken unten jeweils die zutreffenden Ar die richtige Kombination/0P für jede andere).

- (A) eine Erweiterungsklasse von X (falls X eine Klasse ist)
- (B) ein elementarer Datentyp (falls X die zugehörige Wrapper-Klasse ist)
- (C) eine Klasse, die X implementiert (falls X ein Interface ist)

6 Möglichkeiten:
nur A und B sind richtig, nur A und C sind richtig, nur A ist richtig, nur B und C sind richtig, nur B ist richtig, nur C ist richtig

- (D) eine Klasse, von der X direkt oder indirekt erbt (falls X eine Klasse ist)
- (E) ein Interface, das X als abstrakte Klasse erweitert (falls X eine Klasse ist)
- (F) eine Erweiterungsklasse einer Klasse, die X implementiert (falls X ein Interface ist)

6 Möglichkeiten:
nur D und E sind richtig, nur D und F sind richtig, nur D ist richtig, nur E und F sind richtig, nur E ist richtig, nur F ist richtig

13. Analyse von Algorithmen (I1-ID: 7o9glw3115h0)

1. Betrachten Sie folgenden Ausschnitt aus einem größeren Programm:

```
1: for (int i = 0; i < N; i++)
2:     for (int j = i+1; j < N; j++)
3:         for (int k = j+1; k < N; k++)
4:             verarbeite(daten);
```

<Lücke> (<Anzahl der Möglichkeiten>): <Mögl.1>, <Mögl.2>, ...

Die Ausführungszahl $V(N)$ von Zeile 4 bei Problemgröße N soll untersucht werden. Welcher der folgenden Ausdrücke ist am besten für die weitere as geeignet? 1. (9):N*(N-1)*(N-2), N*(N-1)*(N-2)/3, N*(N-1)*(N-2)/6,~N³, ~N³/3, ~N³/6, O(N³), O(N³/3), O(N³/6)

2. Eine Implementation wird zahlreichen Verdopplungstests unterzogen, mit zufällig generierten Eingabedaten. Die Ergebnisse — alle sehr ähnlich — sehen typischer weise so aus:

Problemgröße N	1024	2048	4096	8192	16384	32768	65536	131072	262144	524288	1048576	2097152	4194304	8388608
Zeit (s)	0.00	0.01	0.01	0.02	0.03	0.06	0.14	0.30	0.65	1.24	2.46	4.98	10.02	19.94

Außerdem kann man nachweisen, dass die Ausführungszahl der Kernoperationen bei spezieller Struktur der Eingabe linear mit der Problemgröße N wächst.

Welche dieser Hypothesen über das Laufzeitverhalten erscheint dann am plausibelsten? 2.(8): worst-case quadratisch+best-c. logarithmisch, average-c. linear+best c. quadr., best-c. lin.+avrg.-c. quadr., avrg.-c. quadr.+wrst-c. lin., avrg.-case lin.+best-case lin., worst-c. quadr.+avrg.-c. quadr., best-c. logarithmisch+worst-c. logarithm. avrg.-c. quadr.+worst-c. quadr.

3. Welche dieser Aussagen trifft NICHT zu? 3. (8) Bubblesort ist quadratisch im worst-case, Insertion- und Selection-sort haben asymptotisch gleiche worst-case Laufzeit, Merge-sort aus der Vorlesung ist nicht in-place aber stabil, Binäre Suche erfordert sortierte Eingabe, Quicksort hat auf sortierter Eingabe lineare Laufzeit, Quicksort ist ein in-place Sortierverfahren, Vergleichsanzahl bei Selection-sort hängt nur von Feldlänge ab vergleichsbasierte Sortierverfahren haben mindestens linearithmisches wors-case-Verhalten

14. Eine Bibliothek für Permutationen (I1-ID: l120mlc005h0)

Definition

1. Wir nennen ein `int`-Feld `p` der Länge `n` eine *Permutation*, wenn `p[0]`, ..., `p[n-1]` im Bereich $0, 1, \ldots, n-1$ liegen und jeder Wert genau einmal vorkomm
2. Die Permutation `q` heißt *Inverse* der Permutation `p`, wenn für alle `i` der Wert von `q[p[i]]` gleich `i` ist.

Aufgabe

Schreiben Sie eine Klasse `Permutation` mit den *statischen* Methoden `check(..)` und `invers(..)`, die aus jeder Clientklasse aufgerufen werden können und Folg

- `check(p)` überprüft, ob `p` eine Permutation ist und liefert den Wahrheitswert (**Hinweis**: z.B. zuerst in einer Schleife prüfen, ob die Bereichsbedingung erfü in einem Hilfsfeld `c` die Vorkommen der Werte `p[0]`, ..., `p[n-1]` erfassen)
- `invers(p)` ermittelt und liefert die Inverse von `p`. (Dabei ist bereits gesichert, dass `p` wirklich eine Permutation ist - nicht separat prüfen! **Hinweis**: Feld anl Komponenten so setzen, wie es Definition 2. vorschreibt.)

0 Zeichen zugelassen, Anzahl der eingegebenen Zeichen: **0**

15. Eine einfache Klasse (I1-ID: jr90w0703vg0)

Programmieren Sie eine einfache Klasse `Auto` mit der Angaben zu Autos abgespeichert werden können.
Jedes Auto zeichnet sich durch seinen Typ aus (z.B. "Mercedes") und durch seinen Preis (z.B. 30000).
Die folgenden Operationen sollen durch Methoden/Konstruktoren ermöglicht werden (*wenn nicht anders angegeben*: aufrufbar aus beliebigen Clientklassen):

- Erzeugen einer `Auto`-Instanz unter Angabe des Typs (Typ soll nach Erzeugen des Objekts nicht mehr änderbar sein)
- Erzeugen einer `Auto`-Instanz unter Angabe des Typs und des Preises (Typ soll nach Erzeugen des Objekts nicht mehr änderbar sein)
- Setzen des Preises einer Instanz (*nur von `Auto`-Methoden aus*),
- Abfragen des Preises einer Instanz
- Abfragen der Anzahl der erzeugten Instanzen und der Summe der Preise

VERWENDEN SIE FÜR DIE DATEN BITTE DIESE BEZEICHNER:
`typ`, `preis`, `anzahl`, `gesamt`

0 Zeichen zugelassen, Anzahl der eingegebenen Zeichen: **0**