

**Schriftliche Abiturprüfung
Leistungskursfach Informatik**

- Lösungen zu den Musteraufgaben -

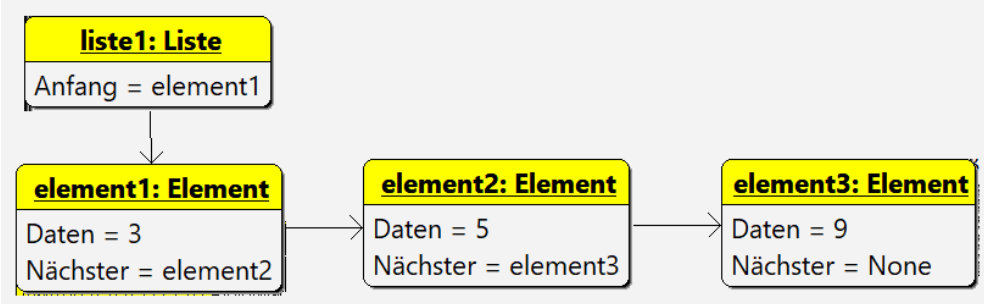
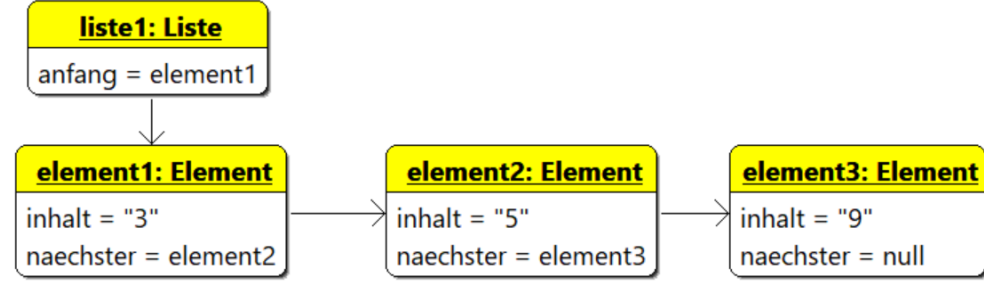
Die nachfolgenden Musterlösungen sind ausführliche Lösungen mit Bewertungshinweisen und stellen somit kein Erwartungsbild dar.

Prüfungsinhalt Teil A

(ohne Nutzung von Tabellen- und Formelsammlung sowie Taschenrechner)

	BE	Lösungsvorschläge und -hinweise
1.1	1	1 BE Antwort 1 angekreuzt 192.213.256.90
1.2	1	1 BE Antwort 2 angekreuzt 12
1.3	1	1 BE Antwort 2 angekreuzt quadratisch
1.4	1	1 BE Antwort 4 angekreuzt abba
1.5	1	1 BE Antwort 2 angekreuzt Es ist nicht möglich, dass zwei Eingabewerte den gleichen Hash besitzen.
2.1	2	Je 1 BE pro Wort für zwei Worte z.B.: aa, baa, babaa...
2.2	2	2 BE Beschreibung Es werden alle Wörter akzeptiert, welche mindestens zwei aufeinanderfolgende a enthalten.
2.3	4	<p>1 BE nichtterminale Symbole, terminale Symbole, Startsymbol, Grammatikschreibweise 3 BE Produktionsregeln – je 1 BE pro Zustand</p> $G = (\{q_0, q_1, q_2\}, \{a, b\}, \{q_0 \rightarrow bq_0, q_0 \rightarrow aq_1, q_1 \rightarrow bq_0, q_1 \rightarrow aq_2, q_1 \rightarrow a, q_2 \rightarrow aq_2, q_2 \rightarrow bq_2, q_2 \rightarrow a, q_2 \rightarrow b\}, q_0)$ <p>oder alternative Darstellungen, z.B.:</p> <p>$G = (N, T, P, s)$ mit $T = \{a, b\}$, $N = \{q_0, q_1, q_2\}$, $s = q_0$ Regeln P: $q_0 \rightarrow bq_0 \mid aq_1$ $q_1 \rightarrow bq_0 \mid a \mid aq_2$ $q_2 \rightarrow aq_2 \mid bq_2 \mid a \mid b$</p> <p>oder</p> <p>$G = (N, T, P, s)$ $N = \{\text{wort, vorn, hinten}\}$ $T = \{a, b\}$ $P = \{$ wort \rightarrow vorn a a hinten vorn \rightarrow a b vorn \mid b \mid b vorn \mid EPSILON hinten \rightarrow a hinten \mid b hinten \mid EPSILON } $s = \text{wort}$</p>

3.1	3	<p>1 BE Nennen von zwei der Anforderungen an die Informationssicherheit</p> <p>z.B.: Vertraulichkeit, Integrität, Authentizität, Verbindlichkeit, Verfügbarkeit erläutern</p> <p>Je 1 BE Begründung der beiden genannten Anforderungen, z.B.:</p> <p><i>Vertraulichkeit</i> Informationen werden nur Berechtigten bekannt -> kein unbefugter Zugriff.</p> <p><i>Integrität</i> Informationen sind richtig, vollständig und aktuell oder dies ist erkennbar nicht der Fall -> die aktuellen Daten müssen von der Behörde bearbeitet werden; der Nutzer muss aktuelle Formulare vorfinden; kein Fremder darf bereits eingegebene Daten verändern können.</p> <p><i>Verfügbarkeit</i> Informationen sind für den Berechtigten stets zugänglich -> wenn der Nutzer weiter an der Dateneingabe arbeiten will, müssen die bereits eingegebenen Daten verfügbar sein.</p> <p><i>Authentizität</i> Der Nutzer muss nachweisen können, dass es seine Daten sind und er muss sicher sein, dass die „Antwortdaten“ von „VOGEL“ stammen.</p> <p><i>Verbindlichkeit</i> „VOGEL“ kann nicht abstreiten, dass ein Dokument aus dem System stammt.</p>
3.2	4	<p>1 BE Szenario für den Einsatz einer Zertifikatsdatei nennen</p> <p>1 BE Szenario für den Einsatz einer Zertifikatsdatei beschreiben</p> <p>2 BE Begründung, wie damit die Informationssicherheit erhöht werden kann</p> <p>Zum Beispiel:</p> <p>Der Nutzer kann sich mit Hilfe dieser Zertifikatsdatei identifizieren, z.B. wenn er Daten hoch- oder runterlädt. Die Informationssicherheit wird erhöht, da der Nutzer zum Datenzugriff zwei unabhängige Informationen benötigt: Passwort (Wissen) und Zertifikatsdatei (Besitz) -> Sicherheitsziel: Authentizität</p> <p>oder</p> <p>Die Zertifikatsdatei dient als private key für den Nutzer. Bei Übermittlung der Daten vom Nutzer an die Firma wird zusätzlich ein Hash-Wert der Daten ermittelt, mit dem private key verschlüsselt und mitgesandt. Somit können die Integrität der übermittelten Daten und Authentizität des Nutzers sichergestellt werden, da der Empfänger den verschlüsselten Hashwert mit dem public key des Nutzers entschlüsselt und bei Übereinstimmung der Hashwerte sicher sein kann, dass das Dokument bei der Übertragung nicht manipuliert wurde und der public key zum Nutzer passt.</p>
3.3	4	<p>2 BE Prozess der Datensicherung beschreiben</p> <p>1 BE Passende Art der Datensicherung nennen</p> <p>1 BE Client- und serverseitige Änderungen nennen</p> <p>Mögliche Lösung (inkl. Art der Sicherung hervorgehoben): Da VOGEL mit wichtigen Daten für Behörden arbeitet, sollte die Sicherung nicht clientseitig (wenig Kontrolle → größere Gefahr des Datenverlusts) sondern serverseitig eingerichtet werden. Es sollte ein tägliches Backup-System nach dem</p>

		<p>RAID-Prinzip eingerichtet werden, in welchem die Daten auf verschiedenen, am besten räumlich getrennten Speichern abgelegt werden. Inkrementelle bzw. differentielle Datensicherung gewährleisten, dass dabei nur Änderungen bzw. Neuerungen gesichert werden um Speicherplatz zu sparen. In regelmäßigen Intervallen (z.B. wöchentlich) sollten ergänzend Komplett Sicherungen durchgeführt werden.</p> <p>Änderungen Serverseite: Einrichten der Hard- und Software für die o.g. Sicherung (z.B. Budgetierung; Dienstleister beauftragen; sicheren Umgang mit sensiblen Daten beachten)</p> <p>Änderungen Clientseite: evtl. Anpassung des Frontend um manuelle Sicherung möglich zu machen; automatische Sicherung empfehlenswert</p>
4.1	2	<p>1 BE korrekte Darstellung der Zahlenkette - es existiert ein Head und die Kette endet auf null/None</p>   <p>1 BE Daten und Zeigerelemente</p>
4.2	3	<p>3 BE alle Teilschritte sind in richtiger Reihenfolge im Pseudocode enthalten</p> <p><u>Möglicher Pseudocode:</u></p> <p>Neues Element (n) erstellen</p> <p>Element (n) mit Nachfolger (n+1) verknüpfen</p> <p>Vorgänger (n-1) mit Element (n) verknüpfen</p>
5	2	<p>2 BE Lösung z.B. über Wahrheitstabelle oder De Morgansche Gesetze</p> <p>$(A \wedge B) = \overline{\overline{A} \vee \overline{B}}$ usw.</p> $\overline{\overline{(A \wedge C)} \wedge (\overline{A} \vee B)} = \overline{\overline{(\overline{A} \vee \overline{C})} \wedge (\overline{A} \vee B)} = \overline{\overline{(\overline{A} \vee \overline{C})}} \vee \overline{(\overline{A} \vee B)} =$ $(A \wedge C) \vee (A \wedge \overline{B}) = A \wedge (C \vee \overline{B}) = A \wedge (\overline{B} \vee C)$ <p>oder</p>

$$\overline{\overline{(A \wedge C)} \wedge (\overline{A \vee B})} = (A \wedge C) \vee \overline{\overline{A \vee B}} = (A \wedge C) \vee (A \wedge \overline{B}) = A \wedge (\overline{B} \vee C)$$

oder

A	B	C	$\overline{A \vee B}$	$\overline{A \wedge C}$	$\overline{(A \wedge C)} \wedge (\overline{A \vee B})$	negiert	D
0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0
0	1	0	1	1	1	0	0
0	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1
1	0	1	0	0	0	1	1
1	1	0	1	1	1	0	0
1	1	1	1	0	0	1	1

6.1

3

1 BE eine Gemeinsamkeit, z.B.

- Formulierung von Projektzielen
- Planvolles Vorgehen bei der Festlegung von Zeitrahmen und Ressourceneinsatz
- in Teamarbeit werden rollenspezifische Aufgaben und Verantwortlichkeiten übernommen

1 BE eine Ähnlichkeit, z.B.

- Projektinformationen werden dokumentiert (Projektplan vs. Sprint-Dokumentation)
- es erfolgt eine Präsentation und Bewertung der (Teil-)produkte (Endbewertung vs. Zwischenbewertungen)

1 BE und ein Unterscheidungspaar, z.B.:

- linear ablaufender vs. iterativer Prozess
- Zentrale Projektleitung vs. Selbstorganisierte Teams
- kaum Einfluss der Stakeholder vs. konstanter Austausch mit Stakeholdern
- einmalige Zieldefinition zu Projektbeginn vs. Kontinuierliche Anpassung der Ziele

der beiden Konzeptformen.

6.2

2

1 BE Auswahl agiler Ansatz

1 BE Begründung, z.B.

Agiler Ansatz aufgrund weniger Vorinformationen, kurzen Zeitraums, nötiger Rücksprachen mit Kunden

6.3	4	<p>2 BE Verwendung projektspezifischer Fachbegriffe (z.B. Sprint, Iteration)</p> <p>1 BE sinnvolle Planung von Überarbeitungen und Feedbackrunden</p> <p>1 BE Projektansatz typische Methoden (z.B. User-Stories, Stand-Up, Sprint-Review)</p> <p><u>Mögliches Beispiel für agiles Konzept:</u></p> <ul style="list-style-type: none"> - Termin und Fragen für Dienstag, z.B. User-Stories (Inhalt, Zielgruppe, Zweck der App) - Teambesprechung, Verteilung Aufgaben, Anlegen KanBan, Priorisierung Aufgaben, Festlegung Sprints und Iterationen (24h mal 5), Wireframe, Moodboard - Dienstagabend Sprint Review - Mittwoch 8:00 Uhr Stand-Up, Besprechung Aufgaben, Priorisierung, Sprint - Mittwoch 18 Uhr Sprint Review - Donnerstag 8:00 Uhr Vorstellung Wireframe/Moodboard, ggf. neue User-Stories und daraus Tasks ableiten, Übergang zu Fast-Prototyping - Stand-Up mit Team und Kundenwünsche besprechen - Aufgaben Priorisieren und Sprint - 19 Uhr Sprint Review - Freitag 9:00 Uhr Stand-Up, Aufgaben besprechen, Aufgaben priorisieren - Freitag 15 Uhr Sprint Review - Montag 8:00 Uhr Stand-Up, werden noch neue Storyboards oder Feedback vom Kunden benötigt? Möchte der Kunde noch etwas klären? -> Falls ja -> Videokonferenz um 9.30 Uhr - Montag 14 Uhr Sprint-Review, Probleme beim Prototyp die behoben werden müssen -> bis 16 Uhr letzte Verbesserungen - Dienstag 8:00 Uhr Vorstellung des Prototyps und Einholung von Feedback
-----	---	--

Prüfungsinhalt Teil B

	BE	Lösungsvorschläge und -hinweise
1.1	9	<p style="color: red;">2 BE Einlesen der Datei 1 BE das Einlesen der Datei findet zeilenweise statt 1 BE Kommentierung und sinnvolle Variablennamen 1 BE speichern in Datenstruktur (z.B. Array / Liste) und Rückgabe 1 BE Implementierung in einer Funktion (namens auslesen) 1 BE Die Funktion wird mit dem Parameter meier_01.csv aufgerufen. 2 BE Begründung für Wahl der gewählten Datenstruktur (z.B. Array/Liste für guten Zugriff, Baum für leichtes Suchen)</p> <p><u>Mögliche Lösung Java:</u> <pre>//import für BufferedReader import java.io.*; //Import für Datenstruktur ArrayList import java.util.ArrayList; public class DateiLesenNeu { private static void auslesen(String fileName) { //leere Datenstruktur erzeugen ArrayList<String> struktur=new ArrayList<String>(); try { //BufferedReader erzeugen und Datei übergeben BufferedReader br = new BufferedReader(new FileReader(fileName)); String line; //solange noch Zeilen vorhanden sind zeilenweise auslesen while ((line = br.readLine()) != null){ //Zeile der Datenstruktur hinzufügen struktur.add(line); } } catch(Exception e) { } } public static void main(String[] args) { //Aufruf der auslesen-Methode und Datei übergeben auslesen("meier_01.csv"); } } </pre></p> <p>Anmerkung: Alternativ auch über Scanner (seit Java 1.5) oder über Stream (seit Java 1.8)</p> <p><u>Mögliche Lösung Python:</u> <pre>def auslesen(name): einkauf=[] # Leeres Array als Datenstruktur # Datei öffnen ohne BOM um Spezialzeichen zu umgehen datei = open(name, 'r', encoding='utf-8-sig') # Zeilenweise einlesen for zeile in datei: einkauf.append(zeile) datei.close() #Datei schließen return(einkauf) # Rückgabe der Struktur print(auslesen('meier_01.csv')) </pre></p>

		<p><u>Variante: mögliche Lösung Python:</u></p> <pre>def auslesen(datei): #Datei öffnen und Inhalt lesen und in der Variable inhalt speichern inhalt = open (datei).read() liste=inhalt.split("") #Unterteilung des Inhalts in Wörter for wort in liste: #zeilenweise print(wort) #Wörter lesen auslesen('meier_01.csv')</pre>
1.2	2	<p>2 BE Lösungsweg und Ergebnis für die theoretische Laufzeitkomplexität Linearer Aufwand $O(n)$, da Anzahl der ausgelesenen Zeilen relevant ist – Nachweis: $O(n)$ für Schleife, ansonsten $O(1)$, anschließend Multiplikationsregel anwenden.</p>
1.3	2	<p>2 BE Vorteile standardisierter Dateiformate nennen Z.B.: Interoperabilität, Einfachheit, platzsparend, Lesbarkeit, Kompatibilität</p>
1.4	3	<p>3 BE Begründen der Vorteile von Datenbanken bei solchen Problemen, z.B. Durch die Trennung von Daten und Programm sind DB deutlich effizienter in der Verwaltung großer Datenmengen</p> <p>DB sind Mehrnutzersysteme → bei großen Datenmengen arbeiten wahrscheinlich mehrere Personen mit den Daten, Anomalievermeidung</p> <p>Leichteres Ändern von Daten in DB</p> <p>Redundanzarmut</p>
2.1	2	<p>2 BE Zum Beispiel: KDNF: $Z = (\bar{A} \wedge B) \vee (A \wedge \bar{B})$</p>
2.2	2	<p>2 BE Grafische Darstellung: Skizziert auf Papier oder erstellt mithilfe einer Modellierungssoftware.</p>

2.3	2	<p>1 BE Ansatz</p> <p>1 BE Vereinfachter boolescher Ausdruck: $X = (\bar{A} \wedge \bar{B}) \vee (\bar{A} \wedge \bar{C})$</p>
3.1	4	<p>4 BE Beschreibung des Datenmanagements</p> <p>Datenmanagement mit Beschreibung einiger Teile des Kreislaufs: Erfassen, Beschaffen, Bereinigen, Analyse, Evaluierung, Visualisieren, Austausch, Löschen und Archivieren von Daten.</p>
3.2	5	<p>1 BE Fremdschlüssel so gewählt, dass alle Entitätsmengen verknüpft sind</p> <p>2 BE Fremdschlüsselzuordnung entsprechend der Transformationsregeln korrekt und sinnvoll erläutert</p> <p>2 BE weitere sinnvolle Attribute begründet angeführt</p> <p>Verschiedene Lösungsvarianten sind möglich. Die Bewertung erfolgt nach logischem Zusammenhang und Korrektheit in Bezug auf die Erläuterungen zu den einzelnen Fremdschlüsseln.</p> <p>Zum Beispiel (Fremdschlüssel hervorgehoben):</p> <p style="padding-left: 40px;">FELSEN (FNr, Name, Einstieg, Schwierigkeitsgrad, Typ, KursNr)</p> <p style="padding-left: 40px;">KLETTERSCHULE (SNr, Bezeichnung, Mail, Tel)</p> <p style="padding-left: 40px;">KURS (KursNr, Kategorie, Niveau, Preis, SNr)</p> <p style="padding-left: 40px;">BUCHUNG (Datum, Uhrzeit, KursNr, KNr)</p> <p style="padding-left: 40px;">KUNDE (KNr, Name, Vorname, Tel, Mail)</p> <p>Die Zuordnung der Fremdschlüssel unter der Annahme, dass</p> <ol style="list-style-type: none"> 1) eine Kletterschule mehrere Kurse anbieten kann → SNr in KURS 2) in einem Kurs verschiedene Felsen bestiegen werden, aber kein Felsen zu mehreren Kursen gehört → KursNr in FELSEN 3) ein Kunde mehrere Kurse buchen kann und an einem Kurs mehrere Kunden teilnehmen können (m:n – Kardinalität) → BUCHUNG als Beziehung und eigene Relation mit Fremdschlüsseln Knr, KursNr <p>Weitere Attribute, z.B. mit KLETTERSCHULE.Adresse, KUNDE.Geschlecht, KUNDE.Kategorie, KUNDE.Rabattstufe, ... möglich, sinnvolle Begründung</p>
3.3	2	<p>2 BE Analyse der Zuordnung, z.B.:</p> <p>Das Attribut Preis ist bei BUCHUNG dynamisch von Datum/Zeit abhängig. Somit kann z.B. ein Kurs mit unterschiedlichen Preisen je nach Auslastung des Kurses angeboten werden.</p>
3.4	7	<p>(I) 2 BE, z.B.</p> <pre>SELECT Name, Vorname, Ausbildungsgrad from Ausbilder where Ausbildungsgrad like 'Trainer C%';</pre> <p>(II) 2 BE, z.B.</p> <pre>SELECT Kategorie, count (distinct (Kurse.K_Nr)) FROM Kurse, leitet WHERE Kurse.K_Nr=leitet.K_Nr AND Datum='08.07.23' GROUP BY Kategorie</pre>

	<p>(III) 3 BE, z.B.</p> <pre>SELECT Name, Vorname, Kategorie, Datum, Uhrzeit FROM Ausbilder inner join (leitet inner join Kurse on leitet.K_Nr=Kurse.K_Nr) on Ausbilder.A_Nr=leitet.A_Nr WHERE Niveau = „Einsteiger“;</pre>
--	---

Prüfungsinhalt Teil C1

	BE	Lösungsvorschläge und -hinweise
1.1	12	<p> 1 BE sinnvolle Variablenbenennung 1 BE Erfassung des Zuges 1 BE Verwaltung der Spieler 1 BE Spielschleife 1 BE Abbruch der Schleife 1 BE Ausgabe des Gewinners 1 BE Anzahl Streichhölzer pro Zug nicht kleiner als 1 1 BE Anzahl Streichhölzer pro Zug nicht größer als 3 1 BE Anzahl gezogener Streichhölzer nicht größer als verfügbare Anzahl an Streichhölzern 1 BE Berechnung der verbleibenden Hölzer 2 BE sinnvolle Kommentare und sauberer Programmierstil </p> <p> <u>Mögliche Lösung Python:</u> #Nim-Spiel sticks = int(input("Bitte geben Sie die Anzahl der Hölzer ein:")) p1 = "Aaron" p2 = "Berta" turn = 0 take = 0 </p> <pre> while True: #Abbruch der Schleife if turn == 0: if sticks <= 0: #Ausgabe des Gewinners print(f"Es gibt keine verfügbaren SpielHölzer mehr, welche {p1} ziehen kann") print(f"{p2} hat gewonnen.") break else: take = int(input(f"{p1} ist am Zug. Es sind noch {sticks} Hölzer vorhanden. Bitte ziehe zwischen 1 und 3 Hölzern:")) #Bedingungen nicht kleiner als 1 und nicht größer als 3 if take > 0 and take < 4: #Bedingung: nicht größer als verfügbare Anzahl if sticks < take: print(f"Es gibt nicht genug Hölzer. Bitte gib eine Zahl zwischen 1 und {sticks} ein.") else: #Berechnung der Hölzer sticks = sticks - take turn = 1 else: print("Bitte geben Sie eine Zahl zwischen 1 und 3 ein.") if turn == 1: if sticks <= 0: print(f"Es gibt keine verfügbaren SpielHölzer mehr, welche {p2} ziehen kann.") print(f"{p1} hat gewonnen.") break else: take = int(input(f"{p2} ist am Zug. Es sind noch {sticks} Hölzer vorhanden. Bitte ziehe zwischen 1 und 3 Hölzern:")) </pre>

```

if take > 0 and take < 4:
    if sticks < take:
        print(f"Es gibt nicht genug Hölzer. Bitte gib eine Zahl zwischen 1 und {sticks} ein.")
    else:
        sticks = sticks - take
        turn = 0
else:
    print("Bitte geben Sie eine Zahl zwischen 1 und 3 ein.")

```

Mögliche Lösung Java:

```

import java.util.*;

public class Nim_imperativ {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Wie viele Hölzer sollen gezogen werden?");
        int anzahl = sc.nextInt();
        int runde=1;
        String spieler="";
        int nimm;
        while (anzahl>0){
            if (runde%2==1){
                spieler = "1";
            }
            else{
                spieler = "2";
            }
            System.out.println("Es existieren noch "+ anzahl + " Hölzer.");
            System.out.println("Spieler "+spieler+ " ist an der Reihe.");
            boolean gueltig=false;
            do {
                System.out.println("Gib an, wie viele Hölzer du ziehen möchtest. ");
                nimm = sc.nextInt();
                if (nimm>0 && nimm<=3 && nimm<=anzahl){
                    gueltig=true;
                }
                else{
                    System.out.println("Ungültig! - Nochmal! ");
                }
            }
            while (!gueltig);
            System.out.println(spieler+" zieht "+nimm);
            anzahl=anzahl-nimm;
            runde=runde+1;
        }
        System.out.println("Es kann kein Holz mehr gezogen werden.");
        System.out.println(spieler + " hat gewonnen!");
    }
}

```

1.2	3	<p>1 BE Ziehender und Sieger sind ersichtlich 1 BE Vollständigkeit der Fälle 1 BE Baum formal richtig dargestellt</p> <p><u>Möglicher Baum:</u></p> <p>1. Zug: Aaron 2. Zug: Berta 3. Zug: Aaron 4. Zug: Berta</p> <p>Berta gewinnt Aaron gewinnt Aaron gewinnt Berta gewinnt Aaron gewinnt Berta gewinnt Berta gewinnt</p>
1.3	2	<p>1 BE Ergebnis: Der Aufruf Baum(3) bewirkt sieben (weitere) rekursive Aufrufe 1 BE Lösungsweg, z.B.</p> <p>Baum(3)</p> <pre> graph TD B3[Baum(3)] --> B2[Baum(2)] B3 --> B1_1[Baum(1)] B3 --> B0_1[Baum(0)] B2 --> B1_2[Baum(1)] B2 --> B0_2[Baum(0)] B1_2 --> B0_3[Baum(0)] B1_1 --> B0_4[Baum(0)] B0_1 --> B0_5[Baum(0)] </pre>

1.4

18

16 BE Implementierung
2 BE Einhaltung der Kriterien zur Codequalität

Mögliche Lösung Python:

```
class Baum:
    def __init__(self, wert, zuege):
        self.wert = wert
        self.zuege = zuege + [wert]
        if wert >= 3:
            self.links = Baum(wert-1, self.zuege)
            self.rechts = Baum(wert-3, self.zuege)
            self.mitte = Baum(wert-2, self.zuege)
        elif wert == 2:
            self.links = Baum(1, self.zuege)
            self.mitte = Baum(0, self.zuege)
        elif wert == 1:
            self.links = Baum(0, self.zuege)
        else:
            if len(self.zuege) % 2 == 0:
                print(self.zuege)

# Programmaufruf
tiefe = input("Gib die Tiefe als ganze Zahl ein.")
baum = Baum(int(tiefe), [])
```

Variante: Mögliche Lösung Python:

```
class Baum:
    def __init__(self, wert, links = None, mitte = None, rechts = None):
        self.wert = wert
        self.links = links
        self.mitte = mitte
        self.rechts = rechts
        if wert >= 3:
            self.links = Baum(wert-1)
            self.rechts = Baum(wert-3)
            self.mitte = Baum(wert-2)
        elif wert == 2:
            self.links = Baum(1)
            self.mitte = Baum(0)
        elif wert == 1:
            self.links = Baum(0)

# Tiefensuche aller Spielverläufe ab Wurzelknoten
def dfs(self, nodes=[]):
    nodes_new = nodes + [self.wert]

    if self.wert == 0:
        if len(nodes_new)%2 == 0:
            print(nodes_new, "Aaron gewinnt")
        else:
            print(nodes_new, "Berta gewinnt")

    if self.links != None:
        self.links.dfs(nodes_new)
    if self.mitte != None :
```

```
self.mitte.dfs(nodes_new)
if self.rechts != None :
    self.rechts.dfs(nodes_new)
```

```
b = Baum(4)
b.dfs()
```

Mögliche Lösung Java:

```
import java.util.ArrayList;

public class Baum {
    private int aktWert;
    private ArrayList<Baum> kindBaeume;
    // Ende Attribute
    public Baum(int startWert) {
        this.aktWert=startWert;
        this.kindBaeume = new ArrayList<Baum>();
        for (int i=1;i<=3;i++) {
            if (this.aktWert>=i) { //Wenn aktWert noch ausreichend groß:
                this.kindBaeume.add(new Baum(this.aktWert-i));
            } //Zug ausführen - sprich: entsprechenden kindBaum erzeugen und in kindBaeume abspeichern
        }
    }
    // Anfang Methoden

    // Rahmenmethode
    public void schreibeSpielverlaeufe() {
        this.schreibeSpielverlauferek("",this.aktWert,0);
    }

    // rekursive Methode
    public void schreibeSpielverlauferek(String verlauf,int vorWert,int tiefe) {
        int spieler = 2- tiefe%2; //aktuellen Spieler berechnen
        if (tiefe!=0) //alle Knoten außer der Wurzel als Zug interpretieren
        {
            verlauf=verlauf+" -> Spieler "+ spieler +" zieht: "+ (vorWert-aktWert);
        } //aktuellen Zug an verlauf anhängen
        if (aktWert==0) {
            if (spieler==1) {
                System.out.println(verlauf+ " und hat damit gewonnen!");
            } //Wenn Endzustand erreicht: Verlauf in Konsole schreiben
        } // end of if

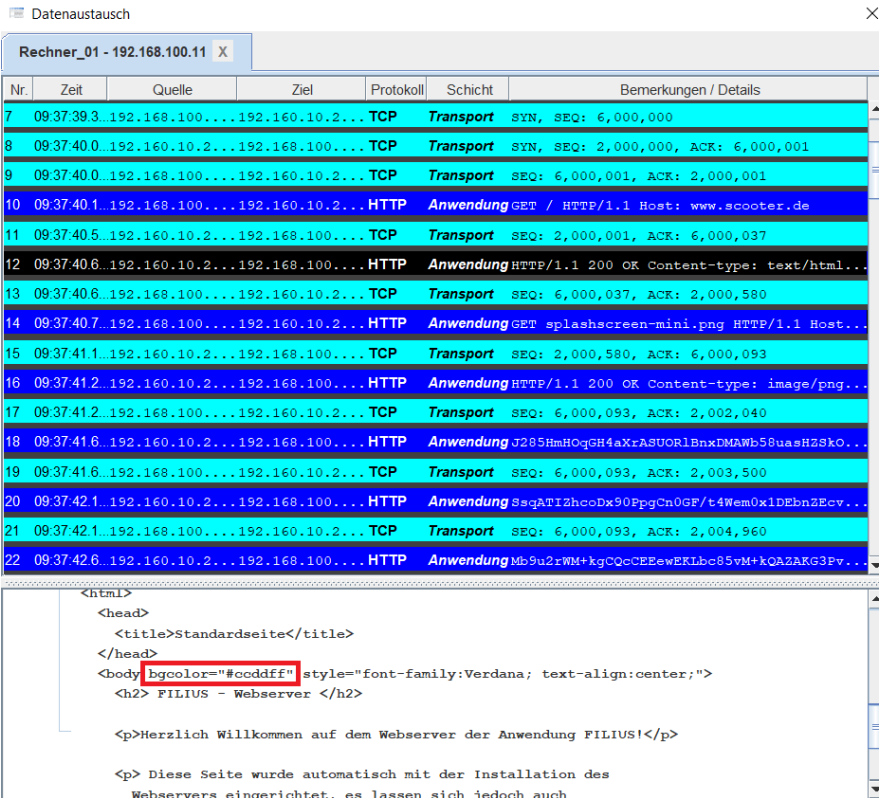
        } else {
            for (int i=0;i<this.kindBaeume.size();i++) {
                this.kindBaeume.get(i).schreibeSpielverlauferek(verlauf,aktWert,tiefe+1);
            } //kein Endzustand erreicht: Endzustand in allen KindKnoten suchen
        }
    }
    // Ende Methoden
}

Baum baum1 = new Baum(4);
baum1.schreibeSpielverlaeufe();
```

1.5	5	<p>3 BE Vergleich</p> <table border="1" data-bbox="352 255 1453 875"> <thead> <tr> <th data-bbox="352 255 906 322">wissensbasierter Ansatz</th> <th data-bbox="906 255 1453 322">datenbasierter Ansatz</th> </tr> </thead> <tbody> <tr> <td data-bbox="352 322 906 875"> <ul style="list-style-type: none"> • Datengrundlage: Expertenwissen • Aufstellen von Regeln durch Entwickler • Mensch entscheidet über Wissen • <i>Beispiele(nicht gefordert)</i> <i>Expertensystem, Entscheidungsbaum, logische Programmierung, A*-Suche</i> </td> <td data-bbox="906 322 1453 875"> <ul style="list-style-type: none"> • Datengrundlage: große Datenmengen, die aufbereitet werden müssen • Trainieren der Daten => Ableiten eines Modells durch das System • mit Testdaten wird das trainierte Modell evaluiert • Maschine entscheidet über Wissen • <i>Beispiele(nicht gefordert): maschinelles Lernen: k-nearest-neighbour, künstliche Neuronale Netze, Naiver Bayes, k-means</i> </td> </tr> </tbody> </table> <p>1 BE Begründung, z.B.</p> <p>Der wissensbasierte Ansatz ist hier besser geeignet, weil das Problem eine explizite Lösung besitzt.</p> <p>1 BE Aufzeigen, z.B.</p> <p>Entscheidungsbaum: Es gibt laut Aufgabenstellung eine feste Regel, nach welcher der perfekte nächste Zug bestimmt werden kann. In Abhängigkeit der noch liegenden Hölzer wird die Regel beschrieben. Der Baum bekommt je möglicher Entscheidung einen eigenen Pfad.</p>	wissensbasierter Ansatz	datenbasierter Ansatz	<ul style="list-style-type: none"> • Datengrundlage: Expertenwissen • Aufstellen von Regeln durch Entwickler • Mensch entscheidet über Wissen • <i>Beispiele(nicht gefordert)</i> <i>Expertensystem, Entscheidungsbaum, logische Programmierung, A*-Suche</i> 	<ul style="list-style-type: none"> • Datengrundlage: große Datenmengen, die aufbereitet werden müssen • Trainieren der Daten => Ableiten eines Modells durch das System • mit Testdaten wird das trainierte Modell evaluiert • Maschine entscheidet über Wissen • <i>Beispiele(nicht gefordert): maschinelles Lernen: k-nearest-neighbour, künstliche Neuronale Netze, Naiver Bayes, k-means</i>
wissensbasierter Ansatz	datenbasierter Ansatz					
<ul style="list-style-type: none"> • Datengrundlage: Expertenwissen • Aufstellen von Regeln durch Entwickler • Mensch entscheidet über Wissen • <i>Beispiele(nicht gefordert)</i> <i>Expertensystem, Entscheidungsbaum, logische Programmierung, A*-Suche</i> 	<ul style="list-style-type: none"> • Datengrundlage: große Datenmengen, die aufbereitet werden müssen • Trainieren der Daten => Ableiten eines Modells durch das System • mit Testdaten wird das trainierte Modell evaluiert • Maschine entscheidet über Wissen • <i>Beispiele(nicht gefordert): maschinelles Lernen: k-nearest-neighbour, künstliche Neuronale Netze, Naiver Bayes, k-means</i> 					

Prüfungsinhalt Teil C2

	BE	Lösungsvorschläge und -hinweise
2.1	6	<p>4 BE Begründung für Beobachtungen</p> <p>(a) durch richtige Konfiguration und intakte Verbindung über den Switch liegen keine Kommunikationsprobleme vor (b) nach erstem Ping: Broadcast von Switch um herauszufinden, welcher Rechner an welchem Port zu finden ist – speichern in Tabelle, nachfolgende Pakete können direkt zugeordnet werden (c) Rechner_03 aufgrund der abweichenden Subnetzmaske in einem anderen Netzwerk. Er ist zwar für Rechner_01 theoretisch erreichbar, kann aber ohne Router (Vermittlungsrechner) nicht auf den Ping antworten -> Timeout (d) Rechner_04 nicht erreichbar, weil er sich mit dieser IP-Adresse nicht im selben Netzwerk befindet – Paket kann ohne Gateway (Vermittlungsrechner) nicht abgesendet werden</p> <p>1 BE Konfiguration des Netzwerks Siehe Lösungsdatei „netzwerk-loesung2-1.fl“</p> <p>1 BE Dokumentation Anpassen der Subnetzmaske von Rechner_03 auf 255.255.0.0 Anpassen der IP-Adresse von Rechner_05 auf z.B. 192.168.75.5 -> Alternative Lösungen denkbar</p>
2.2	2	<p>2 BE je eine BE für jeden Rechner</p> <p>Rechner_02 kann nicht mehr mit Rechner_04 und Rechner_05 kommunizieren, ansonsten keine Einschränkungen Rechner_04 kann nur noch mit Rechner_05 kommunizieren</p>
2.3	13	<p>2 BE Netzwerkstruktur 2 BE Routerkonfiguration (Vermittlungsr.) 2 BE DHCP-Konfiguration 1 BE DHCP bei Clients aktiviert 1 BE E-Mail-Serverkonfiguration 2 BE E-Mailclients funktionstüchtig eingerichtet 1 BE DNS funktionstüchtig eingerichtet 1 BE Webserver eingerichtet 1 BE Webbrowser eingerichtet, Aufruf möglich</p> <p>-> Siehe Lösungsdatei „netzwerk-loesung2-3.fl“</p> <p>Hinweis: Eine stichpunktartige Dokumentation oder Anhand von Screenshots kann bei nicht vollständiger Lösung der Aufgabe für eine Bewertung herangezogen werden.</p>

2.4	3	<p>1 BE Angabe der Farbe 2 BE Nachweis Gültigkeit der Aussage</p>  <p>Angabe der Farbe: z.B. analysieren in Filius der Farbe: #ccddff</p> <p>Nachweis: Im Datenaustausch des Rechners_01 kann man im zweiten gesendeten Protokoll HTTP den html-Code im Klartext sehen. Bei verschlüsselter Übertragung wäre dies nicht möglich.</p>
2.5	6	<p>1 BE Erkennen der Hybridverschlüsselung 1 BE Schlüsselgenerierung (Sessionkey) 1 BE Sicherer Austausch (asymm.) 1 BE symmetrisch Kommunikation</p> <p>Sichere und schnelle Übertragung mittels Hybridverschlüsselung möglich. Schritt 1: Schlüsselgenerierung (Sessionkey) auf Clientrechner Schritt 2: Verschlüsselung des Sessionkeys mithilfe des öffentlichen Schlüssels des Servers (asymmetrische Verschlüsselung) Schritt 3: sichere Übertragung des verschlüsselten Sessionkeys Schritt 4: Symmetrische Kommunikation mittels Sessionkey möglich</p> <p>2 BE Beispiele für Verfahren, z.B. symmetrische Verfahren: AES, TripleDES asymmetrische Verfahren: RSA, ElGamal</p>
2.6	6	<p>1 BE Eingaben/Ausgabe 1 BE Auswahl/Umwandlung geeigneter Datenstrukturen 1 BE Übernahme fester IP-Bestandteile 1 BE Generierung neuer IP-Bestandteile 1 BE Sicherstellung IP-Eindeutigkeit 1 BE Fehlermeldung zu viele Clients</p>

Mögliche Lösung Python:

```
import random
a=int(input('Anzahl der angeschlossenen Clients '))
ip_u_str=str(input('Eingabe IP-Untergrenze '))
ip_o_str=input('Eingabe IP-Obergrenze ')
for k in range(1,len(ip_u_str)):
    ip_u_str=ip_u_str.replace(".",",")
    ip_o_str=ip_o_str.replace(".",",")
ip_u=eval(ip_u_str)
ip_o=eval(ip_o_str)
d=ip_o[3]-ip_u[3]
if d<a:
    print('Es stehen nicht genügend IP-Adressen zur Verfügung')
else:
    ip_l=[]
    i=0
    while i<a:
        ipv4=[]
        for j in range(0,3):
            ipv4.append(ip_u[j])
        el=random.randint(ip_u[3]+1,ip_o[3])
        ipv4.append(el)
        if ipv4 not in ip_l:
            i=i+1
            ip_l.append(ipv4)
    print(ip_l)
```

Mögliche Lösung Java:

```
import java.util.*;
public class IP_Adresse {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Geben Sie die fixen 1. bis 3. Stellen der IP-Adresse mit . getrennt ein: ");
        String fix=sc.next();
        System.out.println("Geben Sie die untere Grenze der 4. Stelle der IP-Adresse ein: ");
        int unten=sc.nextInt();
        System.out.println("Geben Sie die obere Grenze der 4. Stelle der IP-Adresse ein: ");
        int oben=sc.nextInt();
        ArrayList<Integer> menge=new ArrayList<Integer>();
        for (int i=unten;i<=oben ;i++ ) {
            menge.add(i);
        }
        System.out.println("Geben Sie die gewünschte Anzahl an: ");
        int anzahl=sc.nextInt();
        int i=0;
        while (!menge.isEmpty() && i<=anzahl) {
            int zufall=(int)(Math.random()*(menge.size()-1));
            System.out.println(fix+"."+menge.remove(zufall));
            i++;
        }
        if (anzahl>(oben-unten)) {
            System.out.println("Es konnten nur "+(oben-unten+1)+" Adressen vergeben werden.");
        }
    }
}
```

2.7	4	<p>4 BE Beschreibung verschiedener Anpassungen an das Programm</p> <p>Je nach Quelltext aus 2.6 verschiedene Lösungen denkbar.</p> <p>Zum Beispiel:</p> <ul style="list-style-type: none"> - Anpassung der Eingabe an neue Gegebenheit => durch einzelnes Eingeben der Stellen kann auf eine „Zerlegung“ verzichtet werden - Kompliziertere Prüfung „ist Untergrenze kleiner als Obergrenze“, da letzte Stelle bei der Obergrenze kleiner sein kann als bei der Untergrenze. (z.B. 192.168.4.10 kleiner als 192.168.5.5) => zunächst erfolgt eine Prüfung der vorletzten Stelle und nur wenn diese gleich ist, wird die letzte Stelle ausgewertet - Erfassung der verfügbaren Anzahl an IP-Adressen komplizierter (z.B. Adressuntergrenze: 192.168.2.10 und Adressobergrenze: 192.168.4.5 => IP-Adressen von 192.168.2.11 bis 192.168.2.255 = 245 + IP-Adressen von 192.168.3.0 bis 192.168.3.255 = 256 + IP-Adressen von 192.168.4.0 bis 192.168.4.4 = 5 = 506 verfügbare IP-Adressen - Vergabe von Zufallszahlen aus diesem Bereich inkl. Prüfung ob die Zahl schon vorhanden ist - Umrechnung der Zufallszahl auf die Stellen mit geschachtelten Bedingungen <p>Alternative:</p> <ul style="list-style-type: none"> - Vergabe der IP-Adressen in aufsteigender korrekter Reihenfolge - Wenn 255 in der letzten Stelle erreicht – Erhöhung der nächsten Stelle um eins, setzen der letzten Stelle auf 0 - Bei jedem Schritt Verringerung des Zählers [Anzahl] um eins - Wenn Obergrenze erreicht ist, Ausgabe „Es konnten nicht alle IP-Adressen vergeben werden, es fehlen [Anzahl] IP-Adressen“
-----	---	--