



Zentrale Abiturprüfung 2021

Haupttermin

23.04.2021

Profil bildendes Leistungskursfach

Informatik

Fachbereich Informatik

Bearbeitungszeit: 270 Minuten
zusätzliche Rüstzeit: 30 Minuten

Unterlagen für die Schülerinnen und Schüler

Aufgabenstellung

Beschreibung der Ausgangssituation:

Ada ist Informatikerin und bei einer Umweltorganisation aktiv. Dort soll eine App entwickelt werden, mit der der persönliche ökologische Fußabdruck bestimmt werden kann. Dazu soll auch ein Webserver gehören, der auf eine relationale Datenbank zugreift.

Aufgabe 1 – Datenbanksysteme

Folgende Beschreibung hat Ada von der zuständigen Projektgruppe erhalten:

Die Projektgruppe hat eine große Anzahl an Fragen entwickelt. Aus diesen Fragen wird für unterschiedliche Zielgruppen jeweils eine Auswahl zu einem Test zusammengestellt. Jede Frage hat einen Fragetext und bis zu fünf verschiedene Antwortmöglichkeiten. Der Text jeder Antwortmöglichkeit und ihre Position in der Liste der Antworten für die Frage werden gespeichert.

Jede Frage ist einer Kategorie zugeordnet (z. B. „Ernährung“, „Wohnen“, „Transport“) und ist mit einer Punktzahl bewertet, die die Relevanz der Antwort für das Gesamtergebnis beschreibt.

Für jeden Benutzer wird festgehalten, zu welchem Datum er einen Test durchgeführt hat, und für welche Frage er dabei welche Antwortmöglichkeit gewählt hat. Vom Benutzer werden Name und E-Mail-Adresse gespeichert. Jeder Benutzer kann angebotene Tests beliebig oft machen.

Abbildung 1 zeigt eine Beispielfrage.

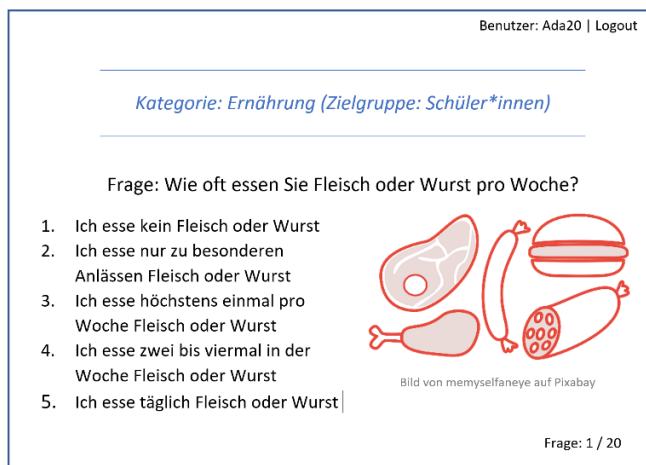


Abb. 1: Beispielfrage

1.1 (25 Punkte)

Erstellen Sie ein ER-Diagramm in MC-Notation, das die oben beschriebene Situation modelliert. Aus der Beschreibung ableitbare Attribute sind zu modellieren.

Hinweis: Die Benutzung von DIA ist verpflichtend.

Bei der Umsetzung eines ER-Modells in ein relationales Schema ist die Einhaltung von Normalformen wichtig, um Anomalien zu vermeiden.

1.2 (6 Punkte)

Erklären Sie die Begriffe „atomar“, „voll funktional abhängig“ und „transitiv abhängig“.



Zur unabhängigen Kommunikation der Mitglieder der Umweltorganisation untereinander soll eine weitere App entwickelt werden, die eine Chat-Funktion bereitstellt. Hierfür wurde das folgende Schema entwickelt:

Benutzer(BenutzerID, Email, Name, Adresse)

Nachricht(NachrichtID, Zeitstempel, Nachrichtentext, GruppeID, Gruppenname, ↑BenutzerID)

Gruppe_Benutzer(↑GruppeID, ↑BenutzerID, GruppenMitgliedSeitDatum, Geburtsdatum)

1.3

(8 Punkte)

Überführen Sie das gegebene Schema in die dritte Normalform.

Um Aussagen zum Klimaschutz mit Zahlen belegen zu können, hat Ada bereits eine Datenbank mit Klimadaten erstellt und mit öffentlich verfügbaren Daten befüllt. Dabei hat sie folgendes Schema verwendet:

Land(LandID, Land)

Kategorie(KategorieID, Kategorie)

CO2Wert(Jahr, ↑LandID, ↑KategorieID, CO2InTonnen)

Eine Mit-Aktivistin schlägt anstelle der Relation CO2Wert die folgende Relation vor:

AlternativeCO2WertRelation(Jahr, ↑LandID, ↑KategorieID, CO2InTonnen)

1.4

(6 Punkte)

Analysieren Sie die Konsequenzen, falls die Relation CO2Wert durch die Relation AlternativeCO2WertRelation ersetzt wird.

Im Folgenden wird das obige Schema mit der Relation CO2Wert verwendet. Die AlternativeCO2WertRelation wurde verworfen.

1.5

(5 Punkte)

Ermitteln Sie eine SQL-Anweisung zur Erstellung der Tabelle CO2Wert.



1.6

(3 Punkte)

Geben Sie eine SQL-Anweisung an, die den gesamten CO₂-Emissionswert des Jahres 2020 ermittelt.

1.7

(6 Punkte)

Ermitteln Sie eine SQL-Anweisung, die alle jährlichen CO₂-Werte Deutschlands in der Kategorie "Flüssige Brennstoffe" absteigend nach Jahr sortiert ausgibt.

1.8

(12 Punkte)

Erstellen Sie eine SQL-Anweisung, die alle Länder ausgibt, die jeweils mindestens 10% zu der Gesamtsumme der weltweiten CO₂-Emission der Jahre 2010 bis 2020 beigetragen haben. Ausgegeben werden sollen das Land und seine gesamte CO₂-Emission des Zeitraums, absteigend sortiert nach der CO₂-Emission.

1.9

(4 Punkte)

Erläutern Sie, was die folgende SQL-Anweisung leistet.

```
SELECT Kategorie, Land, SUM(CO2InTonnen) as Total
FROM Land ld JOIN CO2Wert c ON ld.LandID=c.LandID
JOIN Kategorie k ON k.KategorieID=c.KategorieID
GROUP BY k.KategorieID, ld.LandID
ORDER BY k.Kategorie ASC, Total DESC;
```



Aufgabe 2 – Sortierverfahren und dynamische Datenstrukturen

Sie erhalten im Rahmen der App-Entwicklung zum persönlichen ökologischen Fußabdruck die Aufgabe, unterschiedliche Sortierverfahren zu vergleichen und zu bewerten.

Sortierverfahren

Ein wichtiger Aspekt ist die Stabilität eines Sortierverfahrens.

„Ein stabiles Sortierverfahren ist ein Sortieralgorithmus, der die Reihenfolge der Datensätze, deren Sortierschlüssel gleich sind, bewahrt. Wenn bspw. eine Liste alphabetisch sortierter Personendateien nach dem Geburtsdatum neu sortiert wird, dann bleiben unter einem stabilen Sortierverfahren alle Personen mit gleichem Geburtsdatum alphabetisch sortiert.“

(Quelle: Wikipedia: Stabilität (Sortierverfahren) [19.03.2021])

2.1

(6 Punkte)

Beurteilen Sie für die Sortierverfahren Bubble Sort, Selection Sort und Insertion Sort, ob diese stabil sind.

Zur Veranschaulichung sollen Sie die in der Anlage 1 gegebenen Vornamen mit Bubble Sort und Insertion Sort als Schreibtischtest alphabetisch aufsteigend sortieren.

2.2

(6 Punkte)

Geben Sie für die Sortierverfahren Bubble Sort und Insertion Sort die Reihenfolge der Vornamen in Anlage 1 nach jedem äußeren Schleifendurchlauf an. Beschränken Sie sich dabei auf die ersten drei äußeren Schleifendurchläufe.

Für große Datenmengen sollte ein effizientes Sortierverfahren gewählt werden. Ein geeignetes Verfahren ist der Quick Sort-Algorithmus:

```
void quickSort(int[] feld, int u, int o) {  
    if (u < o) {  
        int p = (u+o)/2;  
        int pn = zerlege(feld,u,o,p);  
        quickSort(feld,u,pn-1);  
        quickSort(feld,pn+1,o);  
    }  
}
```

2.3

(6 Punkte)

Erläutern Sie die vorgegebene Implementierung des Quick Sort.

Hinweis: Auf Details der Implementierung der Methode `zerlege` soll hier nicht eingegangen werden.



Für das Zerlegen gibt es verschiedene Ansätze. Im Internet haben Sie folgende Variante gefunden:

```
int zerlege(int[] feld, int u, int o, int p) {
    int pivot = feld[p];

    swap(feld, p, o);
    int pn = u;
    //(***)

    for (int j = u; j < o; j++) {
        if (feld[j] <= pivot){
            swap(feld, pn, j );
            pn++;
        }
        //(***)
    }

    swap(feld,pn, o);
    //(***)
    return pn;
}

void swap(int[] feld, int x, int y){    //Hilfsmethode zum Vertauschen
    int tmp = feld[x];
    feld[x] = feld[y];
    feld[y] = tmp;
}
```

Um diese Methode zu verstehen, ist es sinnvoll, einen Schreibtischtest zu nutzen.

2.4 (11 Punkte)

Führen Sie einen Schreibtischtest für den Aufruf der Methode `zerlege(feld, 0, 7, 3)` mit der in Anlage 2 gegebenen Belegung des Arrays `feld` durch. Die Belegung des Arrays sowie der Variablen `pn` und `j` sollen immer an den in der Methode `zerlege` mit `(***)` markierten Stellen ausgegeben werden. Verwenden Sie hierzu die Vorlage in Anlage 2.

2.5 (7 Punkte)

Begründen Sie, dass die Implementierung von Quick Sort mit der gegebenen Methode `zerlege` terminiert.



Die Projektleiterin möchte den Aufwand für die vier genannten Verfahren vergleichen, um eine Entscheidung für eines der Verfahren zu treffen.

2.6

(8 Punkte)

Geben Sie die asymptotische Laufzeit in O-Notation für den worst case von Insertion Sort und Quick Sort, den average case von Bubble Sort, Selection Sort und Quick Sort sowie den best case für Insertion Sort, Selection Sort und Quick Sort an. Nutzen Sie dafür die Tabelle in Anlage 3.

Nachdem Sie die Verfahren bewertet haben, zeigt Ihnen die Projektleiterin den Quellcode eines Sortierverfahrens. Sie möchte, dass Sie sich diesen genauer anschauen.

```
private void tausche(int pIndex1, int pIndex2, int[] pDaten) {
    int temp = pDaten[pIndex1];
    pDaten[pIndex1] = pDaten[pIndex2];
    pDaten[pIndex2] = temp;
}

public void sort(int[] pDaten) {
    boolean ret;
    int i=1;
    do {
        ret = false;
        for (int j = 0; j < pDaten.length - i; j++) {
            if (pDaten[j] > pDaten[j + 1]) {
                tausche(j, j + 1, pDaten);
                ret = true;
            }
        }
        i++;
    } while (i < pDaten.length && ret);
}
```

2.7

(8 Punkte)

Analysieren Sie die Anzahl der Feldelementvergleiche im best case und im worst case beim Aufruf der Methode `sort` für ein Feld mit 5 Elementen und allgemein für ein Feld mit n Elementen.

Alternativ wird überlegt, ob zur Speicherung von Objekten bei der Berechnung des ökologischen Fußabdrucks anstelle von Arrays dynamische Datenstrukturen verwendet werden sollten.

2.8

(4 Punkte)

Erklären Sie den Unterschied zwischen Arrays und Listen in Bezug auf Speicherbedarf und Zugriffszeiten.



Die Projektleitung möchte nun, dass Sie einzelne Funktionen implementieren. Dazu händigt Sie Ihnen das folgende Klassendiagramm aus.

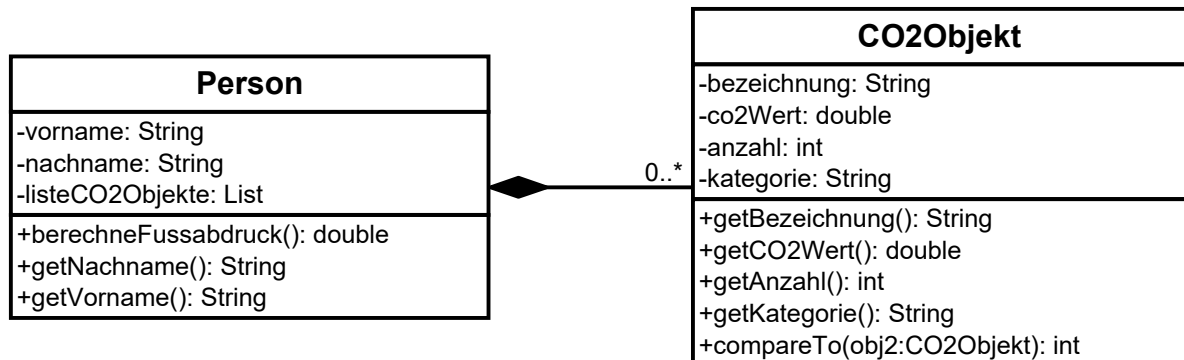


Abb. 2: Klassendiagramm

Die Klasse `Person` besitzt eine Liste aller `CO2-Objekte`. Für eine spätere Analyse der Daten sollen die einzelnen `CO2Objekte` nach Kategorie und innerhalb einer Kategorie nach Bezeichnung sortiert werden.

2.9

(7 Punkte)

Implementieren Sie in der Klasse `CO2Objekt` die Methode `compareTo`, welche die beschriebene Ordnungsrelation realisiert.

Hinweis: In der Java-Dokumentation wird `compareTo` folgendermaßen beschrieben (Auszug): „Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object. [...]“.

Alternativ zur Verwendung einer `compareTo` – Methode kann die oben beschriebene Sortierung nach Kategorie und Bezeichnung auch dadurch erreicht werden, dass ein Sortierverfahren mehrfach angewendet wird. Hierbei wird in jedem Durchlauf nach einem anderen Kriterium sortiert.

2.10

(4 Punkte)

Erklären Sie, welche Anforderungen das Sortierverfahren hierfür erfüllen muss und wie die Durchläufe kombiniert werden müssen.

Der `CO2-Fußabdruck` für eine `Person` wird berechnet, indem über alle ihr zugeordneten `CO2-Objekte` das Produkt von `CO2-Wert` und `Anzahl` aufsummiert wird.

2.11

(8 Punkte)

Implementieren Sie die Methode `berechneFussabdruck` der Klasse `Person`.

Hinweis: In Anlage 4 finden Sie die Dokumentation der Klasse `List`.



Aufgabe 3 – Kryptologie

Die Kommunikation mit dem Web-Server muss verschlüsselt ablaufen.

3.1 (6 Punkte)

Erklären Sie drei Schutzziele der Kryptologie.

3.2 (6 Punkte)

Erklären Sie die monoalphabetische Substitution und die polyalphabetische Substitution unter Angabe je eines Verfahrens.

Die Projektleiterin schickt Ihnen eine mit dem Vigenère-Verfahren verschlüsselte Nachricht.

3.3 (4 Punkte)

Führen Sie die Entschlüsselung der Nachricht „KBEKMOV“ mit dem Schlüssel „INFO“ durch.

Hinweis: In der Anlage 5 finden Sie das Vigenère-Quadrat.

Ein Nachteil des Vigenère-Verfahrens ist es, dass ausschließlich Großbuchstaben verschlüsselt werden können.

3.4 (5 Punkte)

Entwerfen Sie eine Erweiterung des Vigenère-Verfahrens, mit der auch Kleinbuchstaben, Ziffern und Sonderzeichen verschlüsselt werden können.

Für die Kommunikation mit dem Server ist aus Datenschutzgründen eine Transportverschlüsselung vorgeschrieben. Dafür ist es notwendig, einen gemeinsamen Schlüssel mit dem Server zu vereinbaren, wofür das Diffie-Hellman-Verfahren genutzt werden soll.

3.5 (7 Punkte)

Führen Sie das Diffie-Hellman-Verfahren mit den jeweils privaten Informationen $a=5$ und $b=7$, sowie den nicht geheimen Informationen $p=13$ und $g=2$ durch und geben Sie den gemeinsamen Schlüssel an.



Während der Ermittlung eines gemeinsamen Schlüssels mit dem Diffie-Hellman-Verfahren könnten Angreifer alle Informationen abfangen, die zwischen den Kommunikationspartnern ausgetauscht werden. Dazu wird ein interner Test durchgeführt.

3.6 (8 Punkte)

Konstruieren Sie aus den mitgelesenen Werten $g=7$, $p=11$, $A=2$ und $B=4$ den gemeinsamen Schlüssel.

Das Mitlesen der Werte g , p , A und B kann in der Praxis nicht verhindert werden.

3.7 (7 Punkte)

Begründen Sie, warum das Mitlesen der Werte in der Praxis kein Sicherheitsproblem darstellt.

Digitale Signatur

In der App sollen die Ergebnisse mit einer digitalen Signatur unter Verwendung des RSA-Verfahrens versehen werden. Bei der digitalen Signatur wird typischerweise eine Hash-Funktion eingesetzt.

3.8 (6 Punkte)

Erklären Sie, wie eine Hashfunktion beim digitalen Signieren einer Nachricht verwendet wird.

3.9 (6 Punkte)

Erklären Sie, welche Eigenschaften eine Hashfunktion zum Einsatz bei der digitalen Signatur benötigt.



Zu Testzwecken soll die Übertragung einer Nachricht von der App zum Webserver mit den Schlüsseln aus Abbildung 3 unter Verwendung des RSA-Verfahrens digital signiert und verschlüsselt werden. Dabei soll eine geeignete Hash-Funktion h verwendet werden.

	öffentlicher Schlüssel	privater Schlüssel
App	$(N=187, e=3)$	$(N=187, d=107)$
Webserver	$(N=221, e=7)$	$(N=221, d=55)$

Abb. 3: Schlüsselpaare

3.10

(12 Punkte)

Überprüfen Sie rechnerisch, dass bei der verschlüsselten und signierten Übertragung der Klartext-Nachricht $m=44$ von der App zum Webserver die Authentizität der Nachricht verifiziert werden kann. Gehen Sie davon aus, dass $h(44)=17$ gilt.

Auf dem Webserver sollen die Passwörter der Benutzer nicht im Klartext in der Datenbank gespeichert werden. Daher werden nur die Hashwerte der Passwörter abgelegt.

3.11

(4 Punkte)

Erläutern Sie zwei Vorteile der Speicherung von Passwörtern in Form ihrer Hash-Werte.

3.12

(4 Punkte)

Entwerfen Sie eine Angriffsmöglichkeit, falls eine Tabelle mit den Hash-Werten der Passwörter entwendet wird.



Anlagen

Name des Prüflings: _____

Anlage 1 (Aufgabe 2.2)

Bubble Sort

Durchlauf:	Marie	Peter	Martha	Mohamed	Ava	Denys
1						
2						
3						

Insertion Sort

Durchlauf:	Marie	Peter	Martha	Mohamed	Ava	Denys
1						
2						
3						

Anlage 2 (Aufgabe 2.4)

feld[0]	feld[1]	feld[2]	feld[3]	feld[4]	feld[5]	feld[6]	feld[7]	pn	j
5	9	2	7	8	4	2	1		
									/



Name des Prüflings: _____

Anlage 3 (Aufgabe 2.6)

Füllen Sie die freien Felder der Tabelle mit der asymptotischen Laufzeit in O-Notation aus:

Verfahren \ case	worst case	average case	best case
Bubble Sort			
Insertion Sort			
Selection Sort			
Quick Sort			



Anlage 4 (Aufgabe 2.11)

Dokumentation der Klasse `List`

Konstruktor	List() Eine leere Liste wird erzeugt.
Anfrage	boolean isEmpty() Die Anfrage liefert den Wert <code>true</code> , wenn die Liste keine Objekte enthält, sonst liefert sie den Wert <code>false</code> .
Anfrage	boolean hasAccess() Die Anfrage liefert den Wert <code>true</code> , wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert <code>false</code> .
Auftrag	void next() Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt.
Auftrag	void toFirst() Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.
Auftrag	void toLast() Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.
Anfrage	Object getObject() Falls es ein aktuelles Objekt gibt, wird das aktuelle Objekt zurückgegeben, andernfalls gibt die Anfrage den Wert <code>null</code> zurück.
Auftrag	void setObject(Object pObject) Falls es ein aktuelles Objekt gibt und <code>pObject</code> ungleich <code>null</code> ist, wird das aktuelle Objekt durch <code>pObject</code> ersetzt.
Auftrag	void append(Object pObject) Ein neues Objekt <code>pObject</code> wird am Ende der Liste angefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt <code>pObject</code> in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls <code>pObject</code> gleich <code>null</code> ist, bleibt die Liste unverändert.
Auftrag	void insert(Object pObject) Falls es ein aktuelles Objekt gibt, wird ein neues Objekt vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Falls die Liste leer ist und es somit kein aktuelles Objekt gibt, wird <code>pObject</code> in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls es kein aktuelles Objekt gibt und die Liste nicht leer ist oder <code>pObject</code> gleich <code>null</code> ist, bleibt die Liste unverändert.
Auftrag	void concat(List pList) Die Liste <code>pList</code> wird an die Liste angehängt. Das aktuelle Objekt bleibt unverändert. Falls <code>pList</code> <code>null</code> oder eine leere Liste ist, bleibt die Liste unverändert.
Auftrag	void remove() Falls es ein aktuelles Objekt gibt, wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr. Wenn die Liste leer ist oder es kein aktuelles Objekt gibt, bleibt die Liste unverändert.



Anlage 5 (Aufgabe 3.3)

Vigenère-Quadrat

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y



Materialgrundlage

Abbildung 1 steht unter einer freien Lizenz und wurde von memyselfaneye auf Pixabay veröffentlicht, <https://pixabay.com/de/vectors/wurst-hamburger-fleisch-steak-huhn-1791210/> [11. Februar 2021]

Alle anderen Materialien wurden selbst erstellt.

Zugelassene Hilfsmittel

- Wörterbuch der deutschen Rechtschreibung
- Graphikfähiger Taschenrechner (GTR) oder Computeralgebrasystem (CAS)
- Eine Nutzung von Computersystemen ist verpflichtend vorgesehen. Als Programme werden „Dia“ Version 0.97 oder höher und ein Texteditor eingesetzt.

Punktevergabe und Arbeitszeit

Inhaltliche Leistung	225 Punkte
Darstellungsleistung	15 Punkte
Gesamtpunktzahl	240 Punkte

Bearbeitungszeit	270 Minuten
zusätzliche Rüstzeit	30 Minuten