Fachinformatiker/-in Fachrichtung Anwendungsentwicklung FA 234
Entwicklung und Umsetzung von Algorithmen

Lösungsvorschläge: **Lösungsvorschläge sind in der Regel Vorschläge der einreichenden Schulen; sie sind im Wortlaut nicht bindend. Anderslautende, aber zutreffende Antworten sind ebenfalls als richtig zu werten.**

Nur für die Hand des Prüfers! Punkte

Aufgabe 1 **20**

1.1 - CSS – Grid Layout 3
- CSS – Flexboxen
- CSS – Frameworks (z. B. Bootstrap)
- CSS – Eigenschaft position:absolut (zusammen mit Größenangaben)
- HTML – Tabellen

1.2 8

// HTML - Struktur

```
<div class="grid-container">
  <div id="ablauf">  ...</div>
  <div id ="map">    ... </div>
  <div id="auftrag"> ... </div>
  <div id="angebot"> ... </div>
  <div id="buchung"> ... </div>
</div>
```

// CSS - Formatanweisungen

```
.grid-container {
  display: grid;
  grid-template-columns:repeat(4 , 25vw);
  grid-template-rows:  repeat(4 , 25vh);
  }

#ablauf {
  grid-column-start:1;  grid-column-end:5 ;
  grid-row-start:1 ;    grid-row-end:1;
  }

#map {
  grid-column: 1/4;
  grid-row:2/5 ;
  }

#auftrag {
  grid-column-start:4; grid-column-end:5 ;
  grid-row-start:2 ;   grid-row-end:3;
  }

#angebot {
  grid-column-start:4; grid-column-end:5 ;
  grid-row-start:3 ;   grid-row-end:4;
  }

#buchung {
  grid-column-start:4; grid-column-end:5 ;
  grid-row-start:4 ;   grid-row-end:5;
  }
```

1.3 Mit Medienabfragen (Media Queries) können Sie die Darstellung eines Dokuments für verschiedene Ausgabemedien festlegen: 2
- @media screen and (orientation: portrait) { … }
- @media screen and (orientation: landscape) { … }

1.4 Damit ein HTML-Formular barrierefrei genutzt werden kann, benötigt jedes Steuerelement eine Beschriftung, die dem Element eindeutig zugeordnet ist und beispielsweise bei der Nutzung eines Screenreaders die Beschreibung wiedergibt (Web Content Accessibility Guidelines (WCAG)). 3

1.5 4

```
<input type="radio" id="schnell" name="LiefOpt" checked  value="schnell">
<label for="schnell">schnell</label>
<input type="radio" id="guenstig" name="LiefOpt" value="guenstig">
<label  for="guenstig">g&uuml;nstig</label>
```

Aufgabe 2 — 40

2.1 — 10

sexa2dez(sexaWert: String):float
//Beispiel für Übergabewert: 49° 0' 33,228"

lokale Variablen: **var dezWert: float** **var gms: StringArray[]** **var grad, minute, sekunde: float**
gms ← Teile sexaWert an den Leerzeichen in Teilstrings.
grad ← Entferne bei gms[0] das '°'-Zeichen und Umwandlung in **float**
minute ← Entferne bei gms[1] das '-Zeichen und Umwandlung in **float**
sekunde ← Entferne bei gmx[2] das "-Zeichen und Umwandlung in **float**
dezWert ← grad + minute/60 + sekunde/3600
return **dezWert**

2.2 — 25

```
private static double calcPreis(double startLon, double startLat, double zielLon, double zielLat, boolean schnell) {

    /* ----- Dummydaten für Test ----- */

    double preisProKilometerStandard = 3.00;
    double preisZone1Standard = 8.00;
    double preisZone2Standard = 12.00;
    double schnellzustellungStandard = 5.00;

    double zone1 = 4, zone2 = 6;
    double zoneCenterLat = 49.009225, zoneCenterLon = 8.403908;

    /* ------------------------------ */

    double minPreis = 0, distPreis;

    double entfernungStarpunktZone = calcDistance(zoneCenterLon, zoneCenterLat, startLon, startLat);
    double entfernungZielpunktZone = calcDistance(zoneCenterLon, zoneCenterLat, zielLon, zielLat);
    double entfernungStartZiel = calcDistance(startLon, startLat, zielLon, zielLat);

    distPreis = entfernungStartZiel * preisProKilometerStandard;

    if(schnell == true) {
        minPreis = schnellzustellungStandard;
    }

    if (entfernungStarpunktZone <= zone1 && entfernungZielpunktZone <= zone1) {
        if (distPreis < preisZone1Standard) {
            minPreis += distPreis;
        } else {
            minPreis += preisZone1Standard;
        }
    } else if (entfernungStarpunktZone <= zone2 && entfernungZielpunktZone <= zone2) {
        if (distPreis < preisZone2Standard) {
            minPreis += distPreis;
        } else {
            minPreis += preisZone2Standard;
        }
    } else {
        minPreis += distPreis;
    }

    return minPreis;
}

private static double calcDistance(double startLon, double startLat, double zielLon, double zielLat) {

    double distfaktorLAT = 111.13;
    double distfaktorLON = 71.44;

    double entfernung, distLat, distLon;

    distLat = (zielLat - startLat) * distfaktorLAT;
    distLon = (zielLon - startLon) * distfaktorLON;

    entfernung = Math.sqrt(distLat * distLat + distLon * distLon);

    return entfernung;
}
```

2.3 Schülerabhängige Lösungen. Sinnvolle Testdaten wären z. B. 5

- Start- und Zielpunkt sind identisch
- Entfernung zwischen Start- und Zielpunkt kleiner als Zonengröße, Start- und Zielpunkt jedoch außerhalb der Zonen.
- Zielpunkt liegt direkt auf Zonengrenze

Hinweis: Begründung der SuS sollten bei einem fehleranfälligen Grenzfall erklären, weshalb es zu einem unerwarteten Ergebnis kommen könnte.

Aufgabe 3 **30**

3.1 6

- Ein Nachteil nichtnormalisierter Datenbasen ist der erhöhte Speicherbedarf infolge von redundanten Werten.
- Redundante Werte können zu Anomalien (Fehlern) bei der Datenmanipulation führen.
- Attribute, die mehrwertige Daten enthalten lassen sich schlecht auslesen und mit anderen Daten in Beziehung setzen.

3.2 12

Ausgang: Tabelle Kurierfahrten mit ID als Primärschlüssel in Relationen Darstellung

Kurierfahrten (ID,Kunde, ID_Kunde, Datum, Start_LAT_LON, Ziel_LAT_LON, ID_Rad, Typ_Rad, Tarif, startZeit, zielZeit, Fahrzeit, Km_berechnet)

Schritt 0: Beseitigung redundanter Daten laut Aufgabenstellung
Km_berechnet und Fahrzeit sind berechenbar und somit redundant.
Kurierfahrten (ID, Kunde, ID_Kunde, Datum, Start_LAT_LON, Ziel_LAT_LON, ID_Rad, Typ_Rad, Tarif, startZeit, zielZeit)

Schritt 1: 1.Normalform (Atomarität)
Kunde ist nicht atomar -> muss zerlegt werden,
Start_LAT_LON, Ziel_LAT_LON ebenso
(oder in MySQL/MariaDB Verwendung des Datentyps POINT).
Kurierfahrten (ID, K_Zuname, K_Vorname, K_PLZ, K_Ort, K_Strasse, K_HausNr, ID_Kunde, Datum, Start_LAT_LON, Ziel_LAT_LON, ID_Rad, Typ_Rad, Tarif, startZeit, zielZeit)

Schritt 2: 2.Normalform (1NF + voll funktionale Abhängigkeit) Zweite Normalform liegt vor.

Schritt 3: 3.Normalform (1NF + 2NF + keine transitiven Abhängigkeiten)
Kunde (mit allen Attributen) ist nur von ID_Kunde abhängig.
Typ_Rad ist nur von ID_Rad abhängig
Kunde (ID_Kunde, K_Zuname, K_Vorname, K_PLZ, K_Ort, K_Strasse, K_HausNr)
Rad (ID_Rad, Typ_Rad)
Kurierfahrten (ID, *ID_Kunde*, Datum, Start_LAT_LON, Ziel_LAT_LON, *ID_Rad*, Tarif, startZeit, zielZeit)

3.3.1 3

```
SELECT kunde, Km_Berechnet FROM kurierfahrten
ORDER BY Km_Berechnet DESC LIMIT 1 ;
```

3.3.2 3

```
SELECT Tarif , COUNT(Tarif) AS Anzahl
FROM kurierfahrten GROUP BY Tarif ;
```

3.4 6

Ausführung ist einfach (und sollte ohne Taschenrechner möglich sein).
Zuerst wird für jeden Kunden die Summe der gefahrenen km berechnet.
Danach wird absteigend nach dieser Summe sortiert und die ersten drei Zeilen dargestellt.

KundenID	Kilometer
300	31,1
200	26,8
400	23,8