PRG1x & ADE1x Üb. zu Einf. in die Programmierung und Element. Alg. u. Datenstrukt. **WS 20/21, Übung 1**

Abgabetermin: Mi, 14.10.2020

☐ **Gr. 1,** Dr. D. Auer
☒ **Gr. 2,** Dr. G. Kronberger
☐ **Gr. 3,** Dr. S. Wagner

Name Angelos Angelis **Aufwand in h** 9

Punkte _____ **Kurzzeichen Tutor / Übungsleiter** _____ / _____

1. Wöchentliche Arbeitszeit (4 + 4 Punkte)

Gegeben sei eine Folge positiver ganzer Zahlen (wöchentliche Arbeitszeit), die durch die Zahl 0 abgeschlossen ist (die Null gehört nicht mehr zur Zahlenfolge). Entwerfen Sie einen Algorithmus, der die Summe der Überstunden und Minusstunden dieser Folge von wöchentlichen Arbeitszeiten ermittelt und ausgibt. Gehen Sie dabei von einer wöchentlichen Normalarbeitszeit von 40 Stunden aus, d.h. die wöchentlichen Überstunden beginnen ab der 41. Stunde, entsprechendes gilt für die Minusstunden (siehe Beispiele).

Stellen Sie den Algorithmus mittels (a) *Pseudocode* und (b) *Ablaufdiagramm* dar.

Beispiele:

1. Eingabe: `40 42 45 38 40 37 41 0`
   Ausgabe: `Überstunden: 8`
   `Minusstunden: 5`
2. Eingabe: `40 40 40 0`
   Ausgabe: `Überstunden: 0`
   `Minusstunden: 0`

2. Drei Zahlen sortieren (4 + 4 + 4 Punkte)

Entwickeln Sie einen Algorithmus, der drei Zahlen in drei Variablen (z. B. mit den Bezeichnungen *a*, *b* und *c*) einliest und dann die Werte in den drei Variablen in solch einer Reihenfolge ausgibt, dass die Zahlen aufsteigend sortiert sind. Zum Sortieren sollen nur Verzweigungen und Zuweisungen (keine Schleifen) verwendet werden – insbesondere also auch kein „Standard-Sortieralgorithmus“, sollten Sie solche bereits kennen. Stellen Sie Ihren Algorithmus mittels (a) *stilisierter Prosa* und (b) *Ablaufdiagramm* dar und machen Sie (c) einen *Schreibtischtest*, indem Sie Ihren Algorithmus mit der Eingabe 3, 2 und 1 „füttern“.

3. Diskussion: Darstellungsformen (4 Punkte)

Diskutieren Sie die Vor- und Nachteile der Darstellungsformen für Algorithmen, die Sie in Aufgaben 1 und 2 verwendet haben.

Hinweise (diese gelten ab Punkt 2. auch für alle weiteren Übungen):

1. Lesen Sie die organisatorischen Hinweise im Moodle-Kurs.
2. Geben Sie für alle Ihre Lösungen immer eine „Lösungsidee“ an.
3. Dokumentieren und kommentieren Sie Ihre Algorithmen.
4. Bei Programmen: Geben Sie immer auch Testfälle ab, an denen man erkennen kann, dass Ihr Programm funktioniert, und dass es auch in Fehlersituation entsprechend reagiert.

1. Wöchentliche Arbeitszeit **(4 + 4 Punkte)**

Gegeben sei eine Folge positiver ganzer Zahlen (wöchentliche Arbeitszeit), die durch die Zahl 0 abgeschlossen ist (die Null gehört nicht mehr zur Zahlenfolge). Entwerfen Sie einen Algorithmus, der die Summe der Überstunden und Minusstunden dieser Folge von wöchentlichen Arbeitszeiten ermittelt und ausgibt. Gehen Sie dabei von einer wöchentlichen Normalarbeitszeit von 40 Stunden aus, d.h. die wöchentlichen Überstunden beginnen ab der 41. Stunde, entsprechendes gilt für die Minusstunden (siehe Beispiele).

Stellen Sie den Algorithmus mittels (a) *Pseudocode* und (b) *Ablaufdiagramm* dar.

Beispiele:

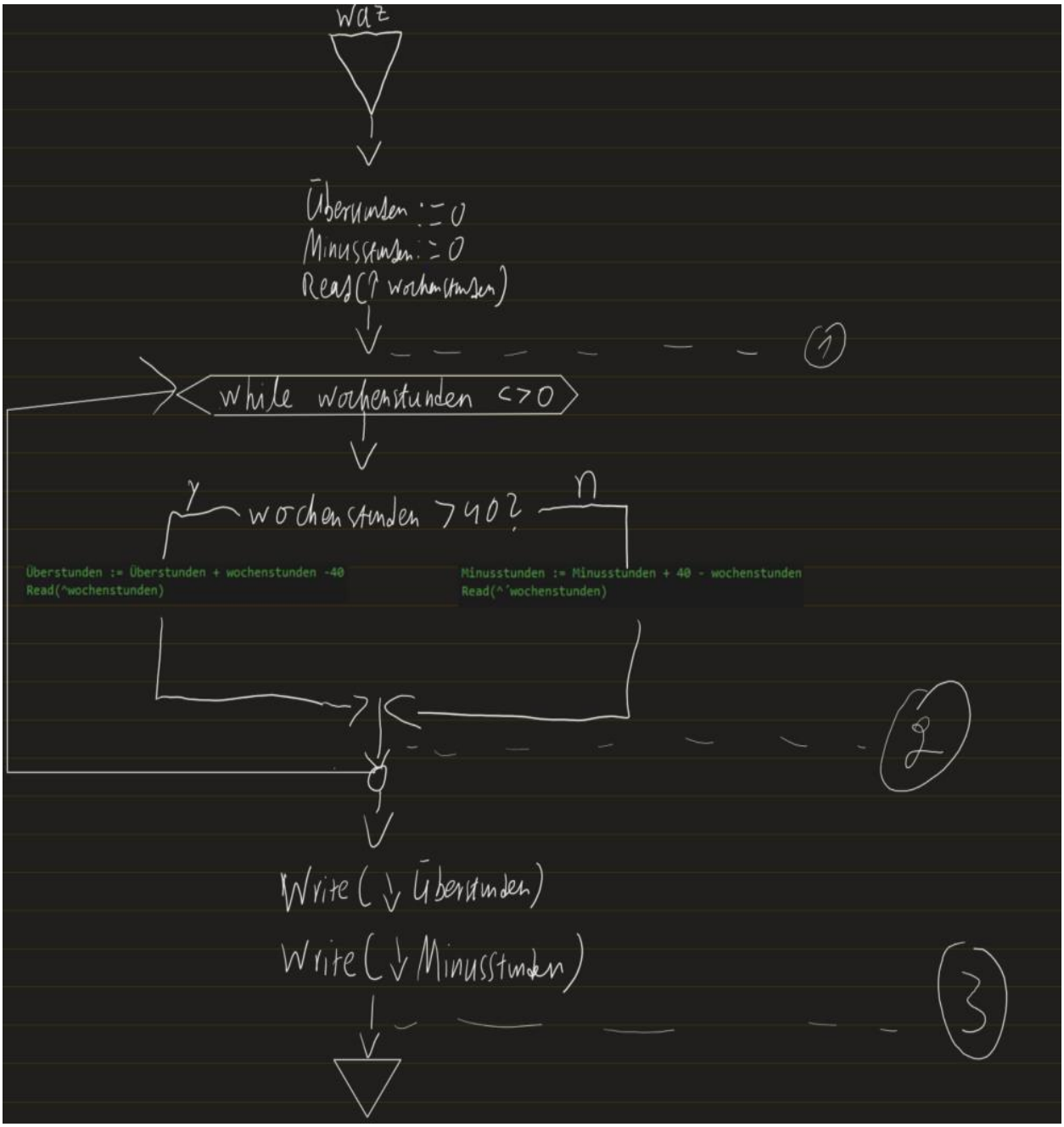
1. Eingabe: `40 42 45 38 40 37 41 0`
   Ausgabe: `Überstunden: 8`
   `Minusstunden: 5`
2. Eingabe: `40 40 40 0`
   Ausgabe: `Überstunden: 0`
   `Minusstunden: 0`

Lösungsidee:

Es müssen Überstunden und Minusstunden ausgerechnet und angezeigt werden. Mithilfe einer Schleife werden die Angaben des Users auf Über und Minus Stunden überprüft. Falls der User eine 0 angibt soll das bedeuten das er mit seinen Angaben fertig ist (Die 0 wird nicht mitgezählt). In der Schleife drin soll dann mit Hilfe der Normalzeit (40 Stunden) überprüft werden ob die angegebenen Wochenstunden des Users Über oder Minusstunden sind. Schließlich werden alle Überstunden und alle Minusstunden zusammengerechnet und ausgegeben.

```
* PSEUDOCODE:
* waz()
* begin
* Überstunden := 0
* Minusstunden := 0
* Read(wochenstunden) //(FUNKTIONIERT NUR MIT NATÜRLICHEN ZAHLEN, ENDE MIT EINGABE 0)
* WHILE wochenstunden <> 0 DO
*      IF wochenstunden > 40 THEN BEGIN
*          Überstunden := Überstunden + wochenstunden -40
*          Read(wochenstunden)
*          END //(*IF)
*      Else BEGIN
*          Minusstunden := Minusstunden + 40 - wochenstunden
*          Read(wochenstunden)
*          END //(*ELSE)
* END
* Write("Überstunden: ", Überstunden)
* Write("Minusstunden: ", Minusstunden)
* end waz
```

Ablaufdiagramm (1)



2. Drei Zahlen sortieren **(4 + 4 + 4 Punkte)**

Entwickeln Sie einen Algorithmus, der drei Zahlen in drei Variablen (z. B. mit den Bezeichnungen *a*, *b* und *c*) einliest und dann die Werte in den drei Variablen in solch einer Reihenfolge ausgibt, dass die Zahlen aufsteigend sortiert sind. Zum Sortieren sollen nur Verzweigungen und Zuweisungen (keine Schleifen) verwendet werden – insbesondere also auch kein „Standard-Sortieralgorithmus", sollten Sie solche bereits kennen. Stellen Sie Ihren Algorithmus mittels (a) *stilisierter Prosa* und (b) *Ablaufdiagramm* dar und machen Sie (c) einen *Schreibtischtest*, indem Sie Ihren Algorithmus mit der Eingabe 3, 2 und 1 „füttern".

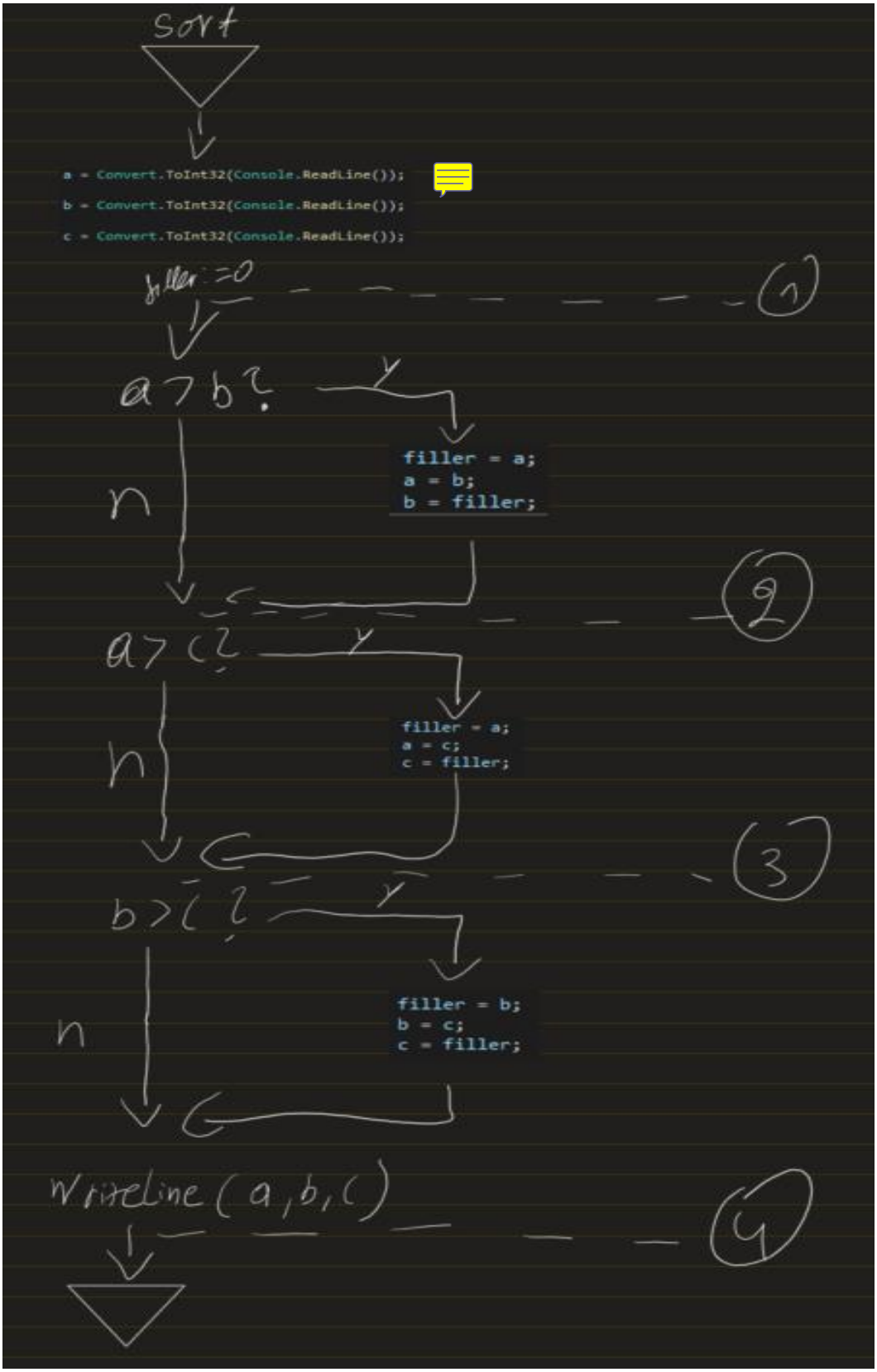
Stilisierte Prosa:

Variablen a,b,c werden anhand der Angaben des Users deklariert. Zudem wird eine weitere Variable deklariert, die als filler benutzt wird. Jetzt mit Hilfe von 3 if Anweisungen werden die 3 Variablen sortiert:

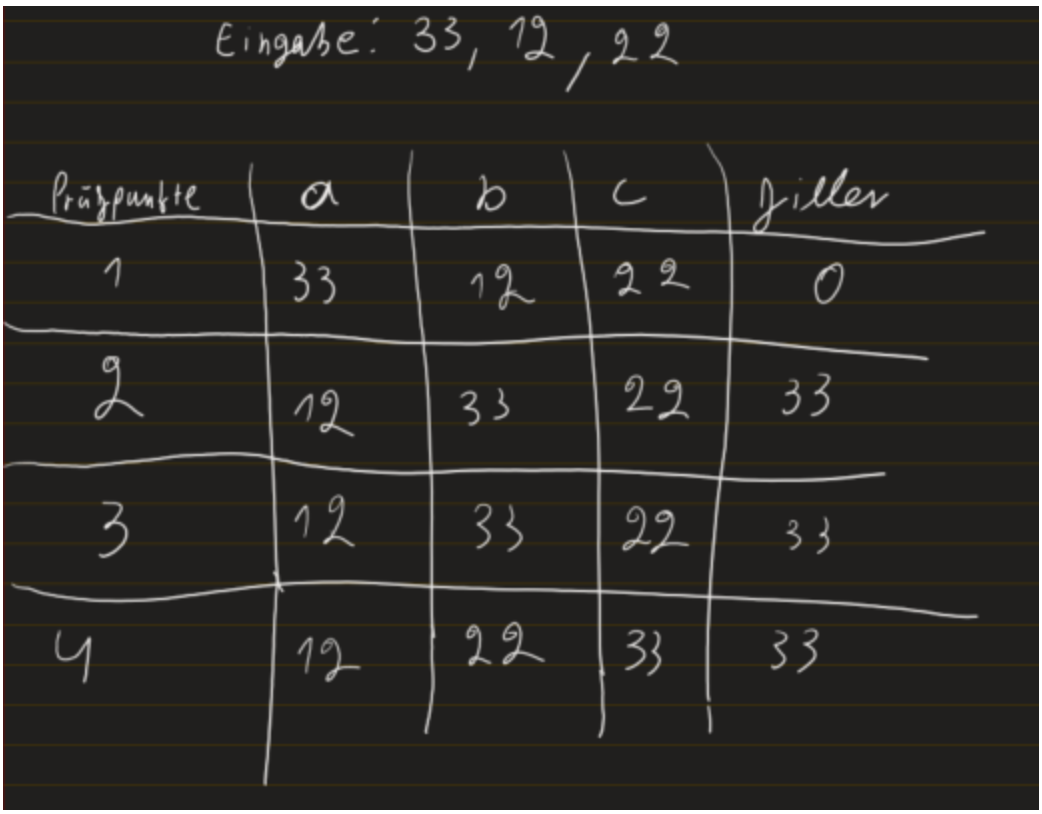
- Die erste if Anweisung Überprüft ob Zahl a größer als Zahl b ist. Falls ja wechseln Zahl a und b Platz mit Hilfe des fillers.
- Bei der zweiten Anweisung wird Überprüft ob Zahl a größer als Zahl c ist. Falls ja wechseln Zahl a und c Platz mit Hilfe des fillers.
- Bei der dritten Anweisung wird Überprüft ob Zahl b größer als Zahl c ist. Falls ja wechseln Zahl b und c Platz mit Hilfe des fillers.

Als letztes werden alle Zahlen ausgegeben

Ablaufdiagramm (2):



Schreibtischtest (2)



3. Diskussion: Darstellungsformen **(4 Punkte)**

Diskutieren Sie die Vor- und Nachteile der Darstellungsformen für Algorithmen, die Sie in Aufgaben 1 und 2 verwendet haben.

Kriterien	Stil.Prosa	Ablaufdiagr.	Pseudocode
Lesbarkeit	-	++	+
Schreibaufwand	--	-/+	+
Eindeutigkeit	--	+	+
Strukturierbarkeit	--	-	+
Datendarstellung	--	-	+
Flexibilität	++	+	+