HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES

20.09.2021

Klausur

Grundlagen der Informatik und Programmierung

Prüfer: Prof. Dr. Malte Weiß

Bewertung

1	2	3	4	5	**Gesamt**	**Note**
52	14	5	20	9	100	

Zum Ablauf der Prüfung

- **Zeit für die Lösung:** 4 Stunden
- **Beginn:** 9:00 Uhr, **Ende:** 13:00 Uhr am 20.09.2021
- **Lösung *wo* hochladen:** in E-Learning im Kursraum der Vorlesung
- **Lösung *wie* hochladen (Form):** Zip-Datei im Format *gdi-exam-VORNAME-NACHNAME-MATRIKELNUMMER.zip*.
  Beispiel: Student *Mathias Bauer* mit Matrikelnummer *10001234* gibt eine Zip-Datei gdi-exam-Mathias-Bauer-10001234.zip ab.
- **Ordnerstruktur:** Jede Aufgabe liegt in einem entsprechenden Unterordner in der Zip-Datei. Siehe aufgabenspezifische Konventionen.
- **Vorlage:** Nutzen Sie die Vorlage, die dieser Klausur beigefügt ist.
- **Nicht erlaubte Hilfsmittel/Hilfe:** Kommunikation digital oder persönlich, direkt oder indirekt mit anderen Menschen (**keine** SMS, WWW-Kommunikation, Telefon, E-Mail, Chat, etc.)
- **Kompilierbarkeit:** Lösungen *müssen* kompilieren, ansonsten werden 50 % der Aufgabenpunkte abgezogen.
- **Fragen:** Für Fragen während der Klausurzeit nutzen Sie bitte das Forum im E-Learning-Bereich der Veranstaltung. Preisgabe von Lösungen oder Ansätzen führt zu Nichtbestehen der Veranstaltung inkl. Fehlversuch.
- **Eidesstattliche Versicherung:** Sie müssen eine eidesstattliche Versicherung abgegeben. Die Klausur gilt als **nicht angetreten**, wenn Sie die eidesstattliche Versicherung nicht unterschrieben einreichen.

Hinweise zur Online-Prüfung

Gemäß § 63 Abs. 5 Satz 1 Hochschulgesetz NRW macht die Hochschule Ruhr West von dem Recht Gebrauch, von den Prüflingen eine Versicherung an Eides Statt (Anlage) zu verlangen und abzunehmen, dass die Prüfungsleistung von ihnen selbstständig und ohne unzulässige fremde Hilfe erbracht worden ist. Von diesem Recht macht die HRW im Rahmen von Online-Prüfungen Gebrauch. Die Prüflinge werden hiermit darüber belehrt, dass die Abgabe einer falschen Versicherung an Eides Statt gemäß § 156 Strafgesetzbuch zu einer Freiheitsstrafe bis zu drei Jahren oder Geldstrafe führen kann. Eine fahrlässige Falschabgabe einer Versicherung an Eides Statt kann gemäß § 161 Strafgesetzbuch zu einer Freiheitsstrafe bis zu einem Jahr oder Geldstrafe führen. Es wird darauf hingewiesen, dass eine Täuschung im prüfungsrechtlichen Sinne vorliegt, sofern nicht der Prüfling selbst, sondern eine andere Person, in ihrem:seinem Namen die Prüfung ablegt. Zudem liegt eine Täuschung im prüfungsrechtlichen Sinne vor, wenn der Prüfling andere als die von dem Prüfer zugelassenen Hilfsmittel nutzt. Das Vorliegen einer Täuschung oder eines Täuschungsversuchs führt dazu, dass die Prüfung mit der Note 5,0 bewertet wird. Zudem kann das Vorliegen einer Täuschung oder eines Täuschungsversuchs gemäß § 63 Abs. 5 Satz 6 Hochschulgesetz NRW sowie § 12 Abs. 3 e) der Einschreibungsordnung HRW zur Exmatrikulation führen.

Sie haben zwei Möglichkeiten, die Eidesstattliche Versicherung abzugeben:

1. Drucken Sie den Text auf der folgenden Seite aus, unterschreiben Sie ihn und scannen/fotografieren Sie ihn ab. Speichern Sie das Foto als Bilddatei im Ordner „Eidesstattliche Versicherung“ in Ihrer Abgabe-ZIP-Datei ab.
2. Kopieren Sie den Text der folgenden Seite in eine Textdatei, setzen Sie ihren Namen und ihre Matrikelnummer darunter und speichern Sie ihn im Ordner „Eidesstattliche Versicherung“ in Ihrer Abgabe-ZIP-Datei ab.

Eidesstattliche Versicherung

Hiermit versichere ich Folgendes an Eides statt:

Ich versichere, dass ich die hier genannte Prüfung selbstständig abgelegt habe.

Zudem versichere ich, dass ich während der hier genannten Prüfung lediglich die von dem Prüfer zugelassenen Hilfsmittel verwendet habe. Ich bestätige, dass ich keinerlei andere Hilfsmittel genutzt habe.

Mir ist bekannt, dass eine eidesstattliche Versicherung eine nach den §§ 156, 161 Strafgesetzbuch (StGB) strafbewehrte Bestätigung der Richtigkeit meiner Erklärung ist. Mir sind die strafrechtlichen Folgen einer unrichtigen Erklärung bekannt.

Nach § 156 StGB wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft, wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung vorsätzlich falsch abgibt. Nach § 161 StGB wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft, wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung fahrlässig falsch abgibt.

Mir ist weiterhin bekannt, dass eine Täuschung im prüfungsrechtlichen Sinne vorliegt, sofern nicht ich selbst, sondern eine andere Person in meinem Namen die Prüfung ablegt. Zudem liegt eine Täuschung im prüfungsrechtlichen Sinne vor, wenn ich andere als die von dem Prüfer zugelassenen Hilfsmittel nutze. Das Vorliegen einer Täuschung oder eines Täuschungsversuchs führt dazu, dass die Prüfung mit der Note 5,0 bewertet wird. Zudem kann das Vorliegen einer Täuschung oder eines Täuschungsversuchs gemäß § 63 Abs. 5 Satz 6 Hochschulgesetz NRW sowie § 12 Abs. 3 e) der Einschreibungsordnung HRW zur Exmatrikulation führen.

Ort, Datum	*Unterschrift der:des Studierenden*

Aufgabe 1: Programmierung (52 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A1 gespeichert werden.

In dieser Aufgabe geht es darum, Inhaltsverzeichnisse aus Dateien in eine Datenstruktur einzulesen und auf dem Bildschirm auszugeben. Eine solche Datei sieht z.B. so aus:

```
Kapitel,Titel,Seite
0,Vorwort,5
1.0,Einleitung,19
1.1,Idee dieses Buches,30
1.2,Beispiele Uebungen und Raetsel,31
1.3,Begleitmaterial zu diesem Buch,32
```

Es handelt sich um eine tabellarische Datei im CSV-Format (Comma Separated Value), in der Spalten durch ein Komma getrennt sind. Die obige Beispieldatei stellt demnach die folgende Tabelle dar:

Kapitel	Titel	Seite
0	Vorwort	5
1.0	Einleitung	19
1.1	Idee dieses Buches	30
1.2	Beispiele Uebungen und Raetsel	31
1.3	Begleitmaterial zu diesem Buch	32

Die beigefügte Datei **chapters.csv** ist eine Beispiel-CSV-Datei, die ein solches Inhaltsverzeichnis darstellt.

Bearbeiten Sie folgenden Teilaufgaben.

Tipp: Wenn Sie eine Teilaufgabe nicht lösen können, sollten Sie trotzdem versuchen, darauf folgende Teilaufgaben zu lösen.

a) Definieren Sie mit **typedef** einen strukturierten Datentyp **Chapter**, der einen Eintrag in einem Inhaltsverzeichnis repräsentiert. Kapitel und Titel sollen als Strings (char-Arrays) umgesetzt werden, die Seitenzahl als **long**-Wert.
(3 Punkte)

b) Definieren Sie mit **typedef** einen strukturierten Datentyp **TableOfContents**, der ein Inhaltsverzeichnis repräsentiert. Er enthält einen Zeiger auf ein Array von Kapiteln (Typ Chapter*) sowie ein Attribut, das die Anzahl der Kapitel darstellt.
(3 Punkte)

c) Schreiben Sie eine Funktion **countDataLines**, die als **Parameter** einen **Dateinamen** einer CSV-Datei in der obigen Form erhält. Die Funktion liest die zum Dateinamen zugehörige Datei ein und zählt die Anzahl der Kapitelzeilen ohne die Kopfzeile. Für das obige Beispiel würde die Funktion die Zahl 5 zurückgegeben.

Tritt ein Fehler auf, wird eine Fehlermeldung ausgegeben und -1 zurückgegeben.

Tipp: Die Anzahl der Zeilen lässt sich über die Anzahl des Zeichens `\n` in der Datei ermitteln.

Hinweis: Schließen der Datei nicht vergessen.
(10 Punkte)

d) Schreiben Sie eine Funktion **createTableOfContents**. Sie erhält als Parameter eine Anzahl von Kapiteleinträgen und gibt ein **TableOfContents**-Wert zurück:

```
TableOfContents createTableOfContents(size_t numChapters)
```

Die Funktion füllt eine TableOfContents-Struktur aus und gibt sie zurück. Dabei setzt sie den Zeiger der Struktur auf ein dynamisch allokiertes Array (Typ Chapter*), dessen Größe dem Wert des Parameters **numChapters** entspricht. Das Attribut für die Anzahl der Kapitel wird ebenfalls gesetzt.

Tritt ein Fehler auf (z.B. wenn Speicherreservierung nicht möglich ist), wird eine Fehlermeldung ausgegeben und eine Struktur zurückgegeben, bei dem der enthaltene Zeiger auf NULL gesetzt wurde.
(6 Punkte)

e) Schreiben Sie eine Funktion **readTableOfContents**. Sie erhält als ersten Parameter den Dateinamen einer CSV-Datei und als zweiten Parameter einen Zeiger auf eine TableOfContents-Struktur, der von der Funktion gesetzt werden soll. Die Funktion soll das Inhaltsverzeichnis aus der übergebenen CSV-Datei einlesen und im Parameter `toc` eintragen.

```
int readTableOfContents(const char* filename, TableOfContents* toc)
```

Im Erfolgsfall gibt die Funktion 1 zurück, sonst 0.

Die Funktion macht folgendes:

1. Sie zählt die Anzahl der Kapiteleinträge der Datei. Nutzen Sie dafür die bereits implementierte Funktion **countDataLines** (siehe Teilaufgabe c).

2. Sie erzeugt ein Inhaltsverzeichnis vom Typ **TableOfContents** an und setzt den obigen `toc` Parameter. Dafür kann die obigen Funktion **createTableOfContents** genutzt werden (siehe Teilaufgabe d).

3. Sie liest die CSV-Datei gemäß dem oben vorgestellten Format ein und füllt das angelegte Inhaltsverzeichnis auf.

Hinweis: Schließen der Datei nicht vergessen.

(12 Punkte)

f) Schreiben Sie eine Funktion **outputTableOfContents**. Als Parameter erhält Sie ein Inhaltsverzeichnis (Typ TableOfContents) und gibt es formatiert aus. Das Kapitel soll rechtsbündig, der Titel linksbündig und die Seitenzahl wiederum rechtsbündig ausgegeben werden, so dass eine Tabellenform entsteht.

Für das obige Beispiel würde die Ausgabe wie folgt aussehen:

```
Kapitel | Titel                                                | Seite
--------+------------------------------------------------------+-------
      0 | Vorwort                                              |     5
    1.0 | Einleitung                                           |    19
    1.1 | Idee dieses Buches                                   |    30
    1.2 | Beispiele Uebungen und Raetsel                       |    31
    1.3 | Begleitmaterial zu diesem Buch                       |    32
```

(8 Punkte)

g) Schreiben Sie eine Methode **safeClearToc**. Sie erhält einen Zeiger auf ein Inhaltsverzeichnis, gibt den Speicher des enthaltenen Kapitel-Arrays frei und setzt die Anzahl der Einträge auf 0.
Hinweis: Stellen Sie sicher, dass ein erneuter Aufruf von safeClearToc mit demselben Zeiger nicht zu einem Absturz führt.
(5 Punkte)

h) Schreiben Sie ein Hauptprogramm, das eine Beispieldatei einliest und ausgibt, so dass alle oben beschriebenen Funktionen mindestens einmal aufgerufen werden.
(5 Punkte)

Aufgabe 2: Zahlendarstellungen in Stellenwertsystemen (14 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A2 gespeichert werden.

In dieser Aufgabe schreiben Sie ein C-Programm, welches eine eingegebene Dezimalzahl in anderen Stellenwertsystemen darstellt und auf der Konsole ausgibt.

a) Lesen Sie eine Dezimalzahl n per Tastaturabfrage im Hauptprogramm ein.

   Schreiben Sie nun eine Funktion **printFormattedNumber**, welche die gerade eingelesene Dezimalzahl n als Übergabeparameter erhält. Die Funktion besitzt keinen Rückgabewert.

   Die Funktion printFormattedNumber gibt die übergebene Dezimalzahl n nun in Oktal- und Hexadezimal-Darstellung auf dem Bildschirm aus.

   Tipp: printf-Dokumentation lesen
   (3 Punkte)

b) Erweitern Sie die Funktion printFormattedNumber so, dass Sie die übergebene Zahl auch im Septenärsystem ausgegeben wird.

   Tipp: Beim Septenärsystem handelt es sich um das Stellenwertsystem mit der Basis 7. Es verwendet also als Basis die Zahlen von 0 bis 6 um alle beliebigen Zahlen darzustellen. Die Dezimalzahl 6 entspräche im Septenärsystem der 6, die Dezimalzahl 7 entspräche der 10.
   (7 Punkte)

c) Erweitern Sie das Hauptprogramm aus Aufgabenteil a), sodass die Zahl n per Kommandozeilenparameter entgegengenommen wird.

   Wird mehr oder weniger als ein Parameter übergeben, wird eine Fehlermeldung ausgegeben.
   (4 Punkte)

Aufgabe 3: Bitfelder (5 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A3 gespeichert werden.

Betrachten Sie die folgenden Struktur:

```
typedef struct {
    int isActive;    // kann 0 oder 1 sein
    long amount;     // ein Wert von 0 bis 15
    int state;       // einer von acht unterschiedlich Zuständen
} SystemState;
```

Schreiben Sie den strukturierten Typ so in ein Bitfeld um, dass nur 1 Byte verbraucht wird, aber trotzdem alle fachlich sinnvollen Werte gespeichert werden können (obige Kommentierung beachten).

Aufgabe 4: Zahlenfolgen (20 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A4 gespeichert werden.

In dieser Aufgabe geht es um Zahlenfolgen, die nach folgendem Gesetz gebildet werden:

1) Wähle eine natürliche Zahl n mit n > 0
2) Ist n gerade, so berechne ganzzahlig n / 2
3) Ist n ungerade, so berechne 3n + 1
4) Wiederhole die Schritte 2) und 3) mit der neu berechneten Zahl, bis die berechnete Zahl gleich 1 ist.

Beispielablauf für das obige Gesetz mit n = 5:

1. n = 5 ist ungerade, daher berechne 3 · 5 + 1. Ergebnis n = 16.
2. n = 16 ist gerade, daher berechne 16 / 2. Ergebnis n = 8.
3. n = 8 ist gerade, daher berechne 8 / 2. Ergebnis n = 4.
4. n = 4 ist gerade, daher berechne 4 / 2. Ergebnis n = 2.
5. n = 2 ist gerade, daher berechne 2 / 2. Ergebnis n = 1.
6. da n = 1: Ende.

Für n = 5 ergibt sich demnach folgende Zahlenfolge: 16, 8, 4, 2, 1

Anderes Beispiel für die Zahlfolge:

Für n = 6 ergibt sich die Folge 3, 10, 5, 16, 8, 4, 2, 1.

Lösen die folgenden Teilaufgaben:

a) Schreiben Sie eine Funktion **calculateSequence**, die für eine übergebene Zahl n die Zahlenfolge des obigen Gesetzes bestimmt und auf dem Bildschirm ausgibt.
(8 Punkte)

b) Schreiben Sie eine Funktion **printSequences**, die für ein eine übergebene Zahl m mit m > 0 von 1 bis m die Zahlenfolgen des obigen Gesetzes bestimmt und auf dem Bildschirm ausgibt.
(2 Punkte)

c) Erweitern Sie das Programm, so dass für ein übergebenes n auf dem Bildschirm zurückgegeben wird, wie oft das obige Gesetz durchgeführt wurde, um bis zur Zahl 1 zu gelangen.
Beispiel: Für n = 6 sind 8 Durchläufe notwendig, um bis zur 1 zu gelangen, für n = 5 sind es 5 Durchläufe.
(2 Punkte)

d) Schreiben Sie eine Funktion **printSequenceChart**, welche die nach dem obigen Gesetz berechnete Zahlenfolge für eine Zahl n als Balkendiagramm auf der Konsole visualisiert. Auf der x-Achse wird die Zahl des aktuellen Rechenschrittes visualisiert.

Auf der y-Achse wird in eine neue Zeile gewechselt, sobald der vorherige Rechenschritt abgeschlossen ist.

Eine beispielhafte Visualisierung für n = 6 (Zahlenfolge 3, 10, 5, 16, 8, 4, 2, 1) sieht wie folgt aus:

```
+
|- - > 3
|- - - - - - - - - - > 10
|- - - -> 5
|- - - - - - - - - - - - - - - - > 16
|- - - - - - - - > 8
|- - - > 4
|- > 2
|> 1
+
```

(8 Punkte)

Aufgabe 5: Zeitmessung (9 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A5 gespeichert werden.

Betrachten Sie den beigelegten Quelltext in der Datei **bartender.c**, der ein simples Gespräch zwischen einem Barmann/Barfrau und einem Kneipenbesucher simuliert.

Der Barmanager möchte gerne möglichst genau wissen, wie viel Zeit vergeht zwischen der Frage nach dem Alter und der Antwort durch den:die Nutzer:in.

Erweitern Sie das Programm, indem Sie eine entsprechende Zeitmessung mit anschließender Ausgabe hinzufügen.