Wirtschaft
Hauptcampus

H O C H
S C H U L E
T R I E R

WINTERSEMESTER 2023/24

Studiengang/Abschluss:	Wirtschaftsinformatik
Prüfungsfach:	Grundlagen der Programmierung
Prüfungsnummer:	***5701***
Prüfer:	Prof. Dr. Andreas Biesdorf
Anzahl der Arbeitsblätter:	**9** Seiten (ohne Deckblätter)
Bearbeitungszeit:	**90** Minuten

Mit Bleistift oder in roter Farbe geschriebene Ausführungen werden nicht gewertet!

Zugelassene Hilfsmittel:

Das im Rahmen der Vorlesung erstellte Hilfsmittel.

Matrikelnummer: ☐☐☐.☐☐☐

Datum der Klausur: …… **Platznummer:** ……

Punkte: _____ von **90** möglichen

Note: _____

Handzeichen des Prüfers: _____

Wirtschaft
Hauptcampus

HOCHSCHULE TRIER

Hinweise zu Klausuren im FB Wirtschaft

- Mit Aufnahme der Prüfung bringen Sie zum Ausdruck, dass Sie sich für prüfungstauglich halten. Ein Rücktritt von der Klausur während der Bearbeitungszeit ist damit nur noch in ganz besonderen Ausnahmefällen möglich.
- Sollten Sie zu Beginn der Bearbeitungszeit prüfungstauglich sein, dann aber während der Bearbeitungszeit prüfungsuntauglich werden, müssen Sie die Klausur abbrechen, Ihre Prüfungsuntauglichkeit den Aufsichtführenden anzeigen und schnellstmöglich, in jedem Fall aber fristgerecht für ein ärztliches Attest sorgen.
- Mit der Abgabe Ihrer Klausur bringen Sie zum Ausdruck, dass Sie sich während der gesamten Bearbeitungszeit für prüfungstauglich gehalten haben. Ein Rücktritt ist dann nicht mehr möglich.
- Jacken und Taschen sind in ausreichender Entfernung vom Platz zu deponieren.
- Die Bearbeitungszeit beginnt erst, wenn alle Klausuren ausgeteilt sind.
- Es sind nur die zugelassenen Hilfsmittel zu verwenden.
- Das Verlassen des Raumes für Toilettengänge während der Klausur ist nur nach Meldung und Bestätigung der Aufsicht gestattet. Die vorübergehende Abwesenheit wird im Protokoll vermerkt.
- Es sind keine anderen als die ausgegebenen Bearbeitungsblätter zu benutzen.
- Auf dem Tisch sind nur Schreibutensilien erlaubt, keine Mäppchen.
- Jeder Teilnehmer muss sich mit einem Studierendenausweis oder Lichtbildausweis ausweisen.
- Geräusche und Störungen jeder Art sind zu vermeiden.
- Die Klausurensätze sind grundsätzlich nicht zu öffnen. Wer den Klausurensatz öffnet, trägt dafür Sorge, dass die Blätter (innerhalb der Bearbeitungszeit) sortiert und neu geheftet werden. Das Risiko des Verlustes von einzelnen Blättern trägt der Studierende!
- Es ist nicht erlaubt, Aufgabenstellungen auf gesondertes Papier abzuschreiben.
- Mobiltelefone, Smartphones und Smartwatches sind nicht erlaubt!
- Alle Studierenden bleiben auf ihren Plätzen und verhalten sich ruhig, bis die Klausuren vollständig eingesammelt sind. Folgen Sie den Anweisungen der Aufsichtsführenden.
- Bei Täuschung oder versuchter Täuschung wird mit „nicht ausreichend" bewertet.

Wirtschaft
Hauptcampus

HOCHSCHULE TRIER

Die Klausur besteht aus 9 Aufgaben zu den in der Lehrveranstaltung behandelten Themenbereichen. Die volle Punktzahl der Klausur beträgt 90 Punkte. Die Punktzahl ist bei jeder Aufgabe mit angegeben.

Aufgabe	**Mögliche Punkte**	**Erreichte Punkte**
A1	10	
A2	10	
A3	10	
A4	10	
A5	10	
A6	20	
A7	10	
A8	10	
A9	25	
Evtl. Bonus		
Summe	115 (von 90)	

Wichtiger Hinweis: Die Klausur bietet Aufgaben für 115 Punkte an, jedoch genügen 90 Punkte für die Erreichung der vollen Punktzahl der Klausur!

Praxisprojekt eingereicht □ Nein □ Ja, in Semester _____

Bitte dokumentieren Sie die Antworten zu allen Aufgaben in der entsprechenden Textbox auf dem Blatt oder der entsprechend benannten Datei (z.B. aufgabe1.txt oder aufgabe7.py) in dem Ordner auf der VM.

Aufgabe 1

1) Was versteht man unter einem Algorithmus und aus welchen drei grundlegenden Bestandteilen besteht dieser? (4 Punkte)

Definition: Bestandteil 1: Bestandteil 2: Bestandteil 3:

2) Nennen Sie **jeweils** ein **Beispiel** für einen festprogrammierten und einen speicherprogrammierten Rechner. In welchem Umfeld können festprogrammierte Rechner einen Vorteil gegenüber speicherprogrammierten Rechnern ausspielen? (3 Punkte)

3) Was bedeutet, dass eine Programmiersprache Turing-vollständig ist? Welche **Auswirkung** hat das auf die Wahl der Programmiersprache **in der Praxis**? (3 Punkte)

Aufgabe 2

Schreiben Sie folgendes Programm mit einem iterativen Ansatz: (10 Punkte)

Berechnen Sie die Summe der ersten n Fibonacci-Zahlen. Der Wert von *n* soll zunächst vom Nutzer abgefragt werden. Die Fibonacci-Zahlen sollen im Rahmen des Programms mit einem iterativen Verfahren bestimmt und ausgegeben werden. Abschließend soll die Summe der ersten *n* Fibonacci-Zahlen berechnet und ausgegeben werden.

Hilfestellung: Die Fibonacci-Zahlenfolge beginnt mit den Zahlen 0 und 1, und jede nachfolgende Zahl ist die Summe der beiden vorherigen Zahlen. Zum Beispiel sind die ersten fünf Fibonacci-Zahlen 0, 1, 1, 2 und 3, und ihre Summe ist 7.

Beispiel 1 (Nutzereingaben in gelb):

```
Wie viele Fibonacci-Zahlen sollen aufsummiert werden? 5
0
1
1
2
3
Summe der ersten 5 Fibonacci-Zahlen: 7
```

Beispiel 2:

```
Wie viele Fibonacci-Zahlen sollen aufsummiert werden? 7
0
1
1
2
3
5
8
Summe der ersten 7 Fibonacci-Zahlen: 20
```

Aufgabe 3

Erstellen Sie ein Programm, welches das Vorkommen von Wörtern in einem beliebigen Text analysiert und die Häufigkeit pro Wort ausgibt. Satzzeichen und Groß-/Kleinschreibung soll hierbei ignoriert werden. (10 Punkte)

Ein Beispiel (Nutzereingaben in gelb):

```
Welcher Text soll analysiert werden? Das ist ein Test. Das ist nur ein Test.

Antwort:
das – 2
ist – 2
ein – 2
test – 2
nur – 1
```

Zur Vereinfachung wird folgende Hilfsfunktion in der Datei `aufgabe03.py` zur Verfügung gestellt, die die Sonderzeichen aus dem Eingabetext entfernt und alle Wörter in Kleinbuchstaben zurückgibt.

```python
import re

def entferne_sonderzeichen(text):
    words = re.sub(r"[^\w\s]", "", text).lower()
    return words

# Beispielaufruf der Funktion
text = "Das ist ein Test. Das ist nur ein Test."
text_ohne_sonderzeichen = entferne_sonderzeichen(text)
print(text_ohne_sonderzeichen)
```

Aufgabe 4

Erstellen Sie ein Programm, welches auf der Kommandozeile eine einfache Zahlenpyramide ausgibt. Die Zahlen laufen von 0 bis 9 und beginnen dann wieder von vorne. In jeder Zeile wird eine zusätzliche Zahl ergänzt. Der Nutzer wird um Angabe der Höhe der Pyramide gebeten und diese dann auf der Kommandozeile ausgegeben. (10 Punkte)

Im Folgenden drei Beispielausgaben für unterschiedliche Höhen:

Zahlenpyramide mit Höhe 3:	Zahlenpyramide mit Höhe 5:	Zahlenpyramide mit Höhe 6:
0 1 2 3 4 5	0 1 2 3 4 5 6 7 8 9 0 1 2 3 4	0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0

Aufgabe 5

Erstellen Sie in der Datei `aufgabe05.py` die folgenden drei Funktionen für einen gegebenen String `str` und rufen diese auf (10 Punkte in Summe; Verteilung siehe unten)

1) `anz_vokale(str)`: (2 Teilpunkte)
   Gibt die Anzahl Vokale im String `str` zurück.
   Hinweis: Vokale sind die Zeichen a, e, i, o, u

   Rückgabewert von `anz_vokale('hstmeanstuasmeanshst')`:
   ```
   6
   ```

2) `zeichen_alphabetisch(str)`: (3 Teilpunkte)
   Gibt die Zeichen eines Strings `str` in alphabetischer Reihenfolge als Liste zurück

   Rückgabe von `zeichen_alphabetisch('hstmeanstuasmeanshst')`:
   ```
   ['a', 'e', 'h', 'm', 'n', 's', 't', 'u']
   ```

3) `laengster_alph_substring(str)`: (5 Teilpunkte)
   Bestimmt den längsten Substring in `str`, bei dem die Buchstaben in alphabetischer Reihenfolge vorkommen.

   Rückgabe `laengster_alph_substring('hstmeanstuasmeanshst')`:
   ```
   'anstu'
   ```

Aufgabe 6

Gegeben ist das folgende *sortierte* Tupel mit den Wörtern der Deutschen Buchstabiertafel:

```
deutsche_buchstabiertafel = ("Anton", "Berta", "Cäsar", "Dora", "Emil",
"Friedrich", "Gustav", "Heinrich", "Ida", "Julius", "Kaufmann", "Ludwig", "Martha",
"Nordpol", "Otto", "Paula", "Quelle", "Richard", "Siegfried", "Theodor", "Ulrich",
"Viktor", "Wilhelm", "Xanthippe", "Ypsilon", "Zacharias")
```

Erstellen Sie nun zwei Funktionen, welche die Position eines der Worte der Buchstabiertafel

1) mit der linearen Suche (8 Punkte):
   `suche_wort_in_buchstabiertafel_linear(suchwort)`
2) mit der binären Suche (10 Punkte):
   `suche_wort_in_buchstabiertafel_binaer(suchwort)`

bestimmt.

Hinweis: Zwei Wörter können in Python einfach mit > / < / == alphabetisch verglichen werden

Beispiel:

```
>>> "Anton" > "Berta"
False
>>> "Anton" < "Berta"
True
>>> "Berta" > "Zacharias"
False
>>> "Eule" > "Computer"
True
```

Sofern ein Wort angegeben wird, welches nicht in der Buchstabiertafel enthalten ist, geben Sie bitte die Position -1 zurück.

Erstellen Sie weiterhin einen einfachen Benchmark: Wie viele Iterationen benötigen Ihre zwei Implementierungen? (2 Punkte)

Beispiel:

```
suche_wort_in_buchstabiertafel_linear ("Dora") = 3
suche_wort_in_buchstabiertafel_linear ("Test") = -1
suche_wort_in_buchstabiertafel_binaer ("Zacharias") = 25
```

Aufgabe 7

Die Fakultät einer nicht negativen ganzen Zahl ist das Produkt aller positiven ganzen Zahlen kleiner oder gleich der Zahl:

$$n! = 1 \cdot 2 \cdot 3 \cdots n = \prod_{k=1}^{n} k$$

Hierbei gilt jedoch

$$0! = 1$$

Einige Beispiele:

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
```

Bitte implementieren Sie die Funktion zur Bestimmung der Fakultät einer Zahl in einem rekursiven Ansatz. (10 Punkte)

Im Folgenden eine Hilfestellung zur Implementierung:

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n-1)!, & n > 0 \end{cases}$$

Aufgabe 8

Das Pascal'sche Dreieck ist folgendermaßen definiert:

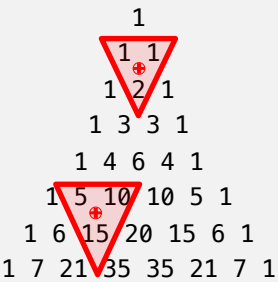
$$p(z,s) = \begin{cases} 1, & s = 1 \vee s = s \\ p(z-1, s-1) + p(z-1, s), & sonst \end{cases}$$

z bezeichnet hier die Zeile und s die Spalte. Hierbei ergibt sich der Wert einer Zahl im Dreieck *anschaulich* immer über die Summe der beiden *grafisch* über der Zahl liegenden Werte in der vorhergehenden Zeile

Hier der Anfang des Dreiecks …

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```

… und hier nochmal in „schön“ 😉



Bitte implementieren Sie die Funktion zur Bestimmung des Werts $p(z,s)$ im Pascal'schen Dreieck mit einem rekursiven Ansatz. Hierbei darf der Nutzer angeben, wie viele Zeilen des Pascal'schen Dreieck er sehen möchte. [10 Punkte]

Hinweis: es ist nicht erforderlich eine besonders ansprechende Visualisierung des Dreiecks zu erstellen. Es genügt eine einfache Ausgabe der Zeilen ähnlich der linken Tabelle.

Aufgabe 9

Eine Bank hat mehrere Konten. Jedes Konto hat eine Kontonummer und einen Kontostand. Bei den Konten gibt es unterschiedliche Typen. Konten können Girokonten, Sparkonten oder Kreditkonten sein. Konten haben folgende Attribute: Kontonummer, Kontostand und Zinssatz. Zusätzlich bietet die Bank verschiedene Dienstleistungen mit Bezug zu den Konten an. Zur Vereinfachung erlauben wir als Dienstleistung lediglich Einzahlungen und Auszahlungen. Jede Dienstleistung hat eine eindeutige Bezeichnung. Die Bank hat auch Kunden. Kunden haben einen eindeutigen Namen. Ein Kunde kann mehrere Konten besitzen und Dienstleistungen der Bank in Anspruch nehmen. Zusätzlich gibt es Mitarbeiter in der Bank. Mitarbeiter haben eine Personalnummer sowie einen Namen. Mitarbeiter sind für die Verwaltung der Konten und die Bereitstellung von Dienstleistungen zuständig und betreuen Kunden. Die Bank ermöglicht es, Kunden und Konten anzulegen und Dienstleistungen (z.B. eine Einzahlung von Geld) darauf anzuwenden.

1] Modellieren Sie die o.g. Zusammenhänge als UML-Modell (10 Punkte)

2) Implementieren Sie die grobe *Struktur* der Klassen Ihres UML-Modells [10 Punkte]

3) Implementieren Sie die für folgende Funktionalität notwendigen Funktionen und führen diese beispielhaft aus: [5 Punkte]
    a. Erstellen eines neuen Kunden mit Ihrem Namen
    b. Anlegen eines Kontos für den gerade erstellten Kunden
    c. Einzahlung von 100€ auf dem gerade angelegten Konto des Kunden

Viel Erfolg bei der Bearbeitung der Aufgaben!