Klausur
Propädeutikum Informatik 2019

21. Juni 2019

Dieses Übungsblatt wird bewertet und bildet die Modulnote für das Propädeutikum Informatik im Sommersemester 2019. Alle Rechnungen sind mit Rechenweg anzugeben. Aufgabe 1,2,3 sind als pdf Datei abzugeben und die Programmieraufgaben in .py Dateien. Die komplette Abgabe ist in einem zip-Archiv bis zum 24.06.2019 23:59 Uhr an fm63byza@studserv.uni-leipzig.de zu senden. Sämtliche Klassen und Methoden, die wir im Propädeutikum erarbeitet haben können genutzt werden. Für jede Programmieraufgabe ist mindestens ein Beispielaufruf im entsprechenden Hauptprogramm anzugeben.

1 Umrechnung zwischen Zahlensystemen

Rechne die folgenden Zahlen vom Dezimal- in das Dualsystem um: 37 und -37. Stelle die Zahlen dabei wie in der Übung als 8-Bit Maschinenzahlen dar.

Rechne die folgene Dualzahl in das Dezimalsystem um: 1011,101.

2 Binäre Suchbäume

Gegeben sei folgende Liste L=[9,5,11,2,1,8,6,10,3,15]. Baue aus L einen Binären Suchbaum auf. Du kannst die Aufgabe mit einem Zeichenprogramm am Computer lösen oder handschriftlich auf Papier und dann eingescannt oder abfotografiert abgeben.

3 Erstellung eines UML-Diagramms

Erstelle ein UML-Diagramm zu folgendem Sachverhalt:

Modelliert werden soll ein Fußballspiel. Eine Person besitzt einen Namen und ein Geburtsjahr. Ein Fußballspieler ist eine Person mit den zusätzlichen Eigenschaften einer Trikotnummer und einer Anzahl bisher geschossener Tore. Eine Fußballmanschaft besitzt einen Namen ein Gründungsjahr, ein Budget, einen Trainer vom Typ Person und 11 Fußballspieler, die in einer Liste gespeichert werden sollen. Ein Fußballspiel besteht aus einer Heimmanschaft und einer Gästemanschaft, einem Schiedsrichter vom Typ Person, und der Anzahl

an Heim- und Gästetoren. Die Klasse Fußballspiel soll außerdem über eine Methode verfügen, welche die Gesamttorzahl zurückgibt und eine Methode, welche die Gewinnermanschaft zurückgibt.

Auf die Angabe von Konstruktoren kann im UML verzichtet werden, die Getter und Setter Methoden für private Attribute sollen aber in jeder Klasse mit angegeben werden. Beachte, dass nur ein UML Diagramm zu erstellen ist. Eine Implementierung in Python ist nicht notwedig. Das UML Diagramm kannst du per Hand zeichnen und dann eingescannt abgeben. Oder du kannst einen online Editor [1] nutzen und dann die Bilddatei speichern.

4 quadratische Gleichung

Schreibe eine Funktion solveEq(a, b, c, d), welche die Lösung einer Quadratischen Gleichung der Form

$$a * x^2 + b * x + c = d$$

berechnet. Welche Formel du zur Berechnung benutzt ist dir überlassen, lass die Funktion das Ergebnis auf dem Bildschirm ausgeben. Sollte die Gleichung kein Ergebnis haben, soll dies auch auf dem Bildschirm ausgegeben werden. Lass in deinem Programm die entspechenden Parameter durch den User über die input Funktion eingeben und gib das entsprechende Ergebnis auf dem Bildschirm aus.

5 showZins

Erweitere die Klasse Sparbuch aus der Übung um eine Methode schowZins(self), welche für das gegebene Guthaben und den gegebenen Zinssatz auf dem Bildschirm ausgibt, wie viele Zinsen man nach 1,2,5 bzw. 10 Jahren bekommen würde. Außer dem Objekt (self), sollen der Methode keine weiteren Parameter übergeben werden.

6 Warteschlangen

In dieser Aufgabe soll eine Warteschlage, zum Beispiel an einer Supermarktkasse simuliert werden. Eine Warteschlange zeichnet sich dadurch aus, dass neue Elemente immer nur an des Ende der Warteschlange angefügt werden können. Ist die Warteschlange voll, also das Maximum an Elementen erreicht, kann kein neues Element gespeichert werden, dann muss eine Fehlermeldung ausgegebene werden. Das nächste zu bearbeitende Element einer Warteschlange ist immer das erste. Im git repo findest du die Klasse Warteschlange.py. Die Warteschlange ist hier intern als Liste abgespeichert und es ist bereits ein Grundgerüst der Klasse vorgegeben. Du hast die Aufgabem die entsprechenden Methoden so zu implementieren, so dass die Warteschlage die Eigenschaften wie oben beschrieben erfüllt.

[1] `http://www.umlet.com/umletino/umletino.html`

7 Perrin-Folge

Schreibe eine Funktion perrinZahl(n): die eine natürliche Zahl n übergeben bekommt und die entspechende Perrin Zahl [2] zurückgibt. Die Perrin Zahlen sind wie folgt definiert:
P(0)=3, P(1)=0, P(2)=2, P(n)=P(n-3)+P(n-2). Schreibe die Funktion perrinZahl in Python. Zusatz: Schreibe außerdem eine Funktion testPerrin, welche die PerrinZahl für die Eingaben 1-100 berechnet, und dann alle Eingaben ausgibt, für die gilt: die Eingabe teilt das Ergebnis.

Hinweis: In einer alten Version der Klausuraufgabe war die Perrin-Folge fälschlicher Weise mit P(n)=P(n-1)+P(n-2) angegeben. Wer die Aufgabe bereits mit der alten Formel gelöst hat, kann hier trotzdem die volle Punktzahl erreichen und muss die Aufgabe nicht neu bearbeiten.

[2] `https://en.wikipedia.org/wiki/Perrin_number`