Wilhelm Büchner Hochschule
Private Fernhochschule Darmstadt

Einsendeaufgabe Typ B **Grundlagen der objektorientierten Programmierung**

Name:		Vorname:	Einsendeaufgabencode: **B-GOP01-XX2-K03**
Straße:		PLZ, Ort:	Korrektor:
Matrikelnummer:		Studiengangsnummer:	Datum:
Name der B-Aufgabe: **B-G O P 0 1 X X**	Variante: **2**	Auflage: **0315K03**	Note:
Bezogene Studienhefte: **GPI01, GPI11A/B, GPI12, GPI13**			Unterschrift:

Bitte reichen Sie Ihre Lösungen über StudyOnline ein. Falls Sie uns diese per Post senden wollen, dann fügen Sie bitte die Aufgabenstellung und den Einsendeaufgabencode hinzu.

1. Der englische Mathematiker John Wallis nutzte folgende Methode zur Berechnung der Zahl Pi:

   $$\frac{\pi}{2} = \frac{2}{1} \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \ldots$$

   Sie heißt nach ihm benannt wallissches Produkt. Programmieren Sie eine C#-Klasse WallisschesProdukt, welche eine Methode enthält, die das wallissche Produkt bis zum *n*-ten Faktor berechnet und zurückgibt. Diese Klasse darf keine Ein-/Ausgabefunktionen wie bspw. `Console.WriteLine()` enthalten. Schreiben Sie ein Hauptprogramm, in dem Sie in einer for-Schleife für alle n von 1 bis 1000 das Produkt ausgeben. Vergleichen Sie die berechneten Ergebnisse mit dem konstanten Wert Math.PI (etwa indem Sie zusätzlich die Differenz Ihres berechneten Wertes zu Math.PI ausgeben). Die Produktformel zur Berechnung des wallisschen Produkts ist nicht zu verwenden. ***15 Pkt.***

2. Gute Passwörter sollten so aufgebaut sein, dass sie nicht leicht durch massives Ausprobieren aufgrund von Wörterbüchern und Zahlenfolgen erraten werden können.

   Ein gutes Passwort sollte zum Beispiel mindestens

   - zwölf Zeichen,
   - zwei Buchstaben,
   - zwei Ziffern,
   - zwei Großbuchstaben und
   - einen Kleinbuchstaben,
   - zwei Sonderzeichen

   enthalten und es sollen sich mindestens

   - eine Ziffer und
   - ein Sonderzeichen, innerhalb des Passwortes befinden, d. h., nicht das erste oder letzte Zeichen sein.

`AxBCD1&&a8cde` erfüllt zum Beispiel diese Kriterien vollständig, `1passw0rtxxx` aber erfüllt nicht alle diese Bedingungen (kein Sonderzeichen und kein Großbuchstabe).

Implementieren Sie eine C#-Klasse `PasswortChecker` mit einer Methode `int isGutesPasswort(string)`, die überprüft, ob ein Passwort obige Kriterien erfüllt oder nicht. Das Passwort soll dabei als Zeichenkette gegeben sein. Für jede erfüllte Bedingung (s. o.) gibt es einen Pluspunkt. Damit sind acht Pluspunkte das Maximum, was von sehr guten Passwörtern erreicht werden kann.

Ein Sonderzeichen soll einfach ein Zeichen sein, das keine Ziffer und kein Unicode-Buchstabe ist. Verwenden Sie keine Methoden der Klasse Char, um einzelne Zeichen zu überprüfen.

20 Pkt.

3. Euro-Banknoten haben eine eindeutige Seriennummer, die aus einem führenden Buchstaben, 10 Ziffern und einer Prüfziffer bestehen. Beispiel: V 0238704003 4 (die Leerzeichen sind zur Übersicht hinzugefügt).

Der führende Buchstabe codiert die nationale Zentralbank (NZB), die den Geldschein in Umlauf gebracht hat. Sie wird NZB-Nummer genannt.

Die Prüfziffer berechnet sich wie folgt:

- Der Buchstabe wird durch seine Position im lateinischen Alphabet ersetzt (hier bei V also 22).
- Es wird die Quersumme der Positionszahl und der 10 Ziffern berechnet (hier im Beispiel 2+2+0+2+3+8+7+0+4+0+0+3 = 31).
- Die Zahl wird mit Rest durch 9 geteilt (hier 4).
- Der Rest wird von 8 subtrahiert. Das Resultat ist die Prüfziffer (hier 4). Es sei denn, es würde 0 dabei herauskommen, dann ist die Prüfziffer 9.

Implementieren Sie eine C#-Methode, die für eine Seriennummer (gegeben als Zeichen in einem char-Feld namens seriennummer) die Prüfziffer berechnet und zurückgibt. Die Seriennummer sei im Feld in umgekehrter Reihenfolge gegeben: seriennummer[0] ist also der Buchstabe, seriennummer[11] die Prüfziffer. Der Buchstabe ist immer als ein Großbuchstabe gegeben.

Fehlerfälle brauchen nicht berücksichtigt zu werden.

Es sind lediglich arithmetische Operatoren und Vergleichsoperatoren zu verwenden, Methoden der Klasse String oder Stringbuffer oder Char dürfen nicht verwendet werden.

20 Pkt.

4. Implementieren Sie jeweils einen rekursiven Algorithmus, der die Summe $a+b$ und das Produkt $a*b$ zweier natürlicher Zahlen rekursiv berechnet. Dabei sind als arithmetische Funktionen lediglich das Addieren von 1 zu einer Zahl oder das Subtrahieren von 1 von einer Zahl erlaubt. Außer `if` sind keine weiteren Kontrollanweisungen erlaubt.

   Zur Implementation der Multiplikation können Sie auf die schon implementierte Addition zurückgreifen.

   Entwerfen Sie außerdem ein Programm, das beide Algorithmen testet.

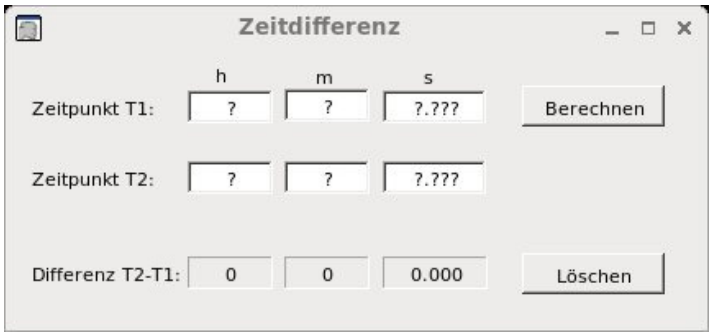
   20 Pkt.

5. Es ist ein Programm zu erstellen, das die Differenz zweier Uhrzeiten T1 und T2 berechnet. T1 und T2 sind in Stunden (h), Minuten (m) und Sekunden (s) einzugeben, wobei die Sekunden reelle Werte sein können.

   Es ist eine Methode `TimeToSec` (mit Rückgabewert) zu schreiben, die die Uhrzeit (h/m/s) in Sekunden umrechnet. Hierbei soll geprüft werden, ob die Uhrzeiten durch gültige Werte repräsentiert werden. Alternativ soll eine Ausnahme ausgelöst werden. Ferner soll eine Methode `SecToTime` (ohne Rückgabewert) hinzugefügt werden, die eine Uhrzeit in Sekunden wieder in Stunden, Minuten und Sekunden konvertiert.

   Die Zeitdifferenz $T2-T1$ ist zu bilden, indem man die beiden Zeiten in Sekunden subtrahiert.

   Erstellen Sie ein Formular in der vergebenen Form:



   Die Schaltfläche `Berechnen` startet die Berechnung der Differenz. Falls die Differenz $T2-T1$ negativ ist, wird beim Ergebnis der Stundenzahl ein Minusvorzeichen vorangestellt.

   Falls die Eingabewerte nicht korrekt sind, bspw. weil keine Zahlen eingetragen werden, soll eine Ausnahme ausgelöst werden. Allerdings sollen Kommata in Punkte (vice versa) bei den Werten der Sekunden umgewandelt werden.

   25 Pkt.

Wichtiger Hinweis

Bitte senden Sie zur Bewertung Ihrer B-Aufgaben die .cs-Quellcodedateien Ihrer Lösungen ein sowie die ausführbaren .exe-Dateien der Programme und alles, was man sonst noch benötigen könnte, um Ihre Programme zu testen. Auch wenn Sie Ihre Programme mit Visual Studio Express erstellt haben, müssen diese auf der Kommandozeile unter dem .NET-Framework bzw. der Mono-Umgebung übersetzbar und ausführbar sein. Gerne können Sie alles in ein ZIP- oder RAR-Archiv packen, um das Hochladen in Study-Online zu vereinfachen.