[…] Felder für Name, Matrikelnummer, das Punkteschema, Datum, etc. (hier ausgelassen)

Allgemeine Randbedingungen und Anforderungen (Beispieltext, kann sich ggf. ändern)

- **Beachten Sie:** Bei jeder Aufgabe gibt es Abzüge, wenn das Programm nicht compiliert, wenn der Code grob ineffizient, kryptisch oder umständlich bzw. unverständlich ist oder wenn Sie einen erfolgreichen Tipp bekommen haben.
- **Generell gilt:** Sollten Sie bei einer Teilaufgabe nicht auf die gefragte Lösung kommen, haben aber eine Alternativlösung, so vermerken Sie das im Code und als Hinweis auf dem Aufgabenblatt! Die Alternative wird mit Abzügen auch gewertet und Sie kommen weiter. Gleiches gilt, falls die Aufgabenstellung an einer Stelle unklar sein sollte. Bitte ebenfalls als Kommentar vermerken.
- **Vorbereitung:** Legen Sie eine Solution mit dem Namen “cs_MATRNR_NAME” an, wobei “MATRNR” durch Ihre Matrikelnummer und “NAME” durch Ihren Nachnamen ohne Umlaute ersetzt ist. Klein oder Großschreibung ist nicht wichtig. In dieser Solution legen Sie ein Projekt je Aufgabe an.
- **Durchführung:** Geben Sie bitte zur Sicherheit Ihren Namen und die Matrikelnummer im Kopf der jeweiligen, von Ihnen bearbeiteten cs-Dateien an.
- **Abgabe:** Sie laden eine zip-Datei namens “cs_MATRNR_NAME.zip” (MATRNR, NAME s.o.) Ihres gesamten Arbeitsverzeichnisses in einen vorgegebenen Ordner im Ilias hoch.

Beispielaufgabe A1

Das Projekt A1 ist eine Konsolenanwendung mit einer statischen Main-Funktion. Es sollen für ein Algebraprojekt natürliche Zahlen untersucht und katalogisiert werden.

Eine natürliche Zahl heißt *abundant*, wenn ihre echte Teilersumme (die Summe aller Teiler ohne die Zahl selbst) größer ist als die Zahl selbst. Ist die Teilersumme dagegen gleich der Zahl, spricht man von einer *vollkommenen* Zahl, ist sie kleiner, so spricht man von einer *defizienten* Zahl.

Beispiele: Echte Teiler von 12 sind 1,2,3,4,6, summiert folglich 16 und da 16>12 gilt, ist 12 abundant. Die Zahl 6 ist wegen ihrer echten Teiler 1,2,3 und 6=1+2+3 vollkommen.

Sie sollen zunächst die Zahlen 1-100 mit einer *geeigneten* parallelen Strategie Ihrer Wahl unter Verwendung von 4 Threads (oder Tasks) untersuchen und jede Zahl entsprechend ihres jeweiligen Typs (abundant, vollkommen oder defizient) in eine der drei Ergebnismengen (set_A, set_V, set_D) thread-sicher einteilen.

a) Implementieren Sie zunächst eine Funktion checkZahl, die eine übergebene Zahl testet, ob sie abundant (Rückgabe 0), vollkommen (Rückgabe 1) oder defizient (Rückgabe 2) ist. Schreiben Sie dazu geeigneten Testcode.

Punkte: [_ / 5+2]

b) Implementieren Sie Ihre parallele Strategie und teilen Sie die Zahlen 1-100 in die drei Mengen set_A, set_V und set_D thread-sicher ein.

Punkte: [_ / 10]

c) Filtern Sie mit einem LINQ-Ausdruck die Zahlen aus set_A aus, die durch 10 teilbar sind und generieren Sie eine Ergebnismenge, die aus anonymen Objekten mit einem Feld nummer für die Zahl und einem Feld sum für die jeweilige Summe besteht.
Geben Sie diese Menge aus und nutzen Sie bei der Ausgabe String-Interpolation.

Punkte: [_ / 5+3]