Aufgabe 1: Die To-Do Liste

Thema: Strings verwalten

1. Erstellen Sie eine `List<string>` namens `aufgaben`.
2. Fügen Sie "Lernen", "Einkaufen", "Schlafen" hinzu.
3. "Schlafen" ist nicht so wichtig. Entfernen Sie es wieder (`Remove`).
4. Fügen Sie "Programmieren" ganz am Anfang der Liste ein (Index 0).
5. Geben Sie alle Aufgaben mit einer `foreach`-Schleife auf der Konsole aus.

Note:

- Gehen Sie herum und prüfen Sie auf `using System.Collections.Generic;`.

Aufgabe 2: Der Notenschnitt

Thema: Rechnen mit dynamischen Daten

1. Erstellen Sie eine `List<double>` für Noten.
2. Fügen Sie folgende Noten hinzu: `1.3`, `2.7`, `1.0`, `4.0`.
3. Berechnen Sie den Durchschnitt:
   - Erstellen Sie eine Variable `summe = 0`.
   - Iterieren Sie mit `foreach` durch die Liste und addieren Sie jede Note zur Summe.
   - Teilen Sie die Summe durch `liste.Count`.
4. Geben Sie den Durchschnitt auf der Konsole aus.

Note:

- Vorsicht bei der Division, falls Count 0 ist (hier nicht der Fall, aber theoretisch möglich).

Aufgabe 3: Das Login-System

Thema: Prüfen von Keys

1. Erstellen Sie ein `Dictionary<string, string>` namens `userDaten`.
   - Key: Benutzername, Value: Passwort.
2. Fügen Sie User hinzu: ("admin", "1234") und ("gast", "0000").
3. Simulieren Sie einen Login:
   - Legen Sie `eingabeUser = "admin"` an.
   - Prüfen Sie mit `ContainsKey`, ob der User existiert.
   - **Wenn ja:** Geben Sie das Passwort aus `userDaten[...]` aus.
   - **Wenn nein:** Geben Sie "User unbekannt" aus.

Aufgabe 4: Die Preisliste

Thema: Werte verändern

1. Erstellen Sie ein `Dictionary<string, double>`, das Produkte (Key) und Preise (Value) speichert.
2. Fügen Sie ein: "Apfel" (0.99), "Brot" (2.50), "Kaffee" (4.00).
3. Der Benutzer möchte wissen, was ein "Brot" kostet.
   - Greifen Sie auf das Element "Brot" zu und speichern Sie den Preis in einer Variable.
4. Der Preis von "Kaffee" steigt. Überschreiben Sie den Wert für den Key "Kaffee" mit `4.50`.
5. Geben Sie zur Kontrolle alle Preise mit einer Schleife aus.

Note:

- Werte überschreiben geht über den Indexer: `liste["Key"] = NeuerWert;`. `Add` würde hier crashen.

Aufgabe 5: Der E-Mail Filter

Thema: Einzigartigkeit (HashSet)

1. Erstellen Sie ein `HashSet<string>` namens `emails`.
2. Fügen Sie "a@test.de" hinzu.
3. Fügen Sie "b@test.de" hinzu.
4. Fügen Sie "a@test.de" **noch einmal** hinzu.
5. Geben Sie `emails.Count` auf der Konsole aus.
   - *Frage: Ist das Ergebnis 2 oder 3?*

Note:

- Ergebnis muss 2 sein. Das Duplikat wird ignoriert.

Aufgabe 6: Gemeinsame Interessen

Thema: Schnittmengen (Intersect)

1. Erstellen Sie ein HashSet `gruppe1` mit {"Sport", "Musik", "Reisen"}.
2. Erstellen Sie ein HashSet `gruppe2` mit {"Musik", "Lesen", "Sport"}.
3. Nutzen Sie `gruppe1.IntersectWith(gruppe2)`, um herauszufinden, was **beide** Gruppen gemeinsam haben.
4. Geben Sie die verbleibenden Elemente von `gruppe1` aus.

Note:

- Lösung: "Sport" und "Musik".

Aufgabe 7: Der Drucker

Thema: Queue (FIFO)

1. Erstellen Sie eine `Queue<string>` namens `druckWarteschlange`.
2. Fügen Sie "Dokument1.pdf" und "Dokument2.pdf" hinzu (`Enqueue`).
3. Der Drucker will wissen, was als nächstes kommt, aber noch nicht drucken.
   - Nutzen Sie `Peek` und geben Sie das Ergebnis aus.
4. Der Druckvorgang startet. Holen Sie das erste Dokument mit `Dequeue` aus der Schlange und geben Sie "Drucke: ..." aus.

Aufgabe 8: Wort umdrehen

Thema: Stack (LIFO)

1. Erstellen Sie einen `Stack<char>`.
2. Nutzen Sie eine `foreach`-Schleife über den String `"HALLO"`, um jeden Buchstaben einzeln auf den Stack zu legen (`Push`).
3. Nutzen Sie eine zweite Schleife (z.B. `while(stack.Count > 0)`), um die Buchstaben mit `Pop()` wieder vom Stack zu holen.
4. Geben Sie die Buchstaben direkt hintereinander aus (`Console.Write` statt `WriteLine`).
   - *Ergebnis sollte "OLLAH" sein.*

Note:

- Klassiker-Aufgabe um LIFO zu verstehen.