Einführung in die Programmierung
Anhang A: Versionsverwaltung mit Git

1) In dieser Aufgabe machen Sie die ersten praktischen Übungen mit der Versionsverwaltung Git.
   Legen Sie zu nächst auf GitHub.com einen Account an. Legen Sie dann ein neues (leeres) Repository namens `TestRepo` mit den Standardeinstellungen an.

   Installieren Sie nun den GitHub Desktop Client auf Ihrem Rechner, zu beziehen unter https://desktop.github.com/ (nur für Windows, für andere OS einen alternativen Client nutzen). GitHub Desktop verlangt beim Start den GitHub-Server und Ihre Benutzerkennung. Wählen Sie hier GitHub.com und Ihre Kennung.
   Nun können Sie unter "`File -> Clone a Repository`" lokale Kopien ("Klons") beliebiger Repositories erstellen. Klonen Sie Ihr auf GitHub erstelltes Repository `TestRepo` und wählen Sie als lokalen Pfad z.B. "`C:\GitHub\TestRepo`".

   Wenn Sie parallel mit der Kommandozeile arbeiten wollen, fügen Sie zunächst den Pfad zur Datei "`git.exe`" der Umgebungsvariablen "`Path`" auf Ihrem PC hinzu. Standardmäßig ist das z.B. ein Pfad wie der folgende:
   "`C:\Users\<Benutzer>\AppData\Local\GitHubDesktop\app-2.6.0\resources\app\git\mingw64\bin`"
   (anpassen und `<Benutzer>` ersetzen). Öffnen Sie ein neues Kommandozeilenfenster und prüfen Sie, ob der Aufruf des Kommandos `git` erfolgreich ist. Das Kommando `git config --list` sollte dann neben anderen Informationen auch Ihre Benutzerkennung anzeigen.

   Um die Kommandozeile zu nutzen, müssen Sie noch einen Editor für die Commit-Messages festlegen. Am einfachsten wählen Sie den Windows-Standardeditor, indem Sie folgendes Kommando absetzen:
   `git config --global core.editor "notepad"`

   Wechseln Sie nun in das Verzeichnis "`C:\GitHub\TestRepo`" und legen ein neues Unterverzeichnis "`HelloGit`" an. Wechseln Sie in dieses Verzeichnis und erstellen zwei Dateien, "`Hello.c`" und "`Greeter.c`". Prüfen Sie mittels "`git status`", was sich geändert hat. Fügen Sie diese neuen Dateien nun mittels "`git add`" zum Index hinzu. Prüfen Sie wieder mittels "`git status`", was beim nächsten Commit passieren würde.
   Fügen Sie Ihre Änderungen nun mittels "`git commit`" zu Ihrem lokalen Repository hinzu. Prüfen Sie wieder mittels "`git status`" den aktuellen Stand und verfolgen Sie die Änderungen auch im GitHub Desktop UI nach.

   Ändern Sie nun die Datei "`Hello.c`" mit Hilfe eines Editors und speichern Sie diese. Prüfen Sie mittels "`git status`" den aktuellen Stand. Fügen Sie Ihre gemachten Änderungen mittels "`git commit -a`" Ihrem lokalen Repository hinzu und schauen Sie sich diese auch im GitHub Desktop UI an (-> `History` Tab).

   Alles, was Sie bis jetzt über die Kommandozeile durchgeführt haben, lässt sich auch bequem über das UI erledigen. Üben Sie diese Schritte nochmals anhand

der Datei "`Greeter.c`", nun mit Hilfe des UIs.

Nun ist es an der Zeit, alle Änderungen in Ihrem lokalen Repository mit dem zentralen Server GitHub.com abzugleichen. Wählen Sie hinzu "`Push origin`" bzw. "`Repository -> Push`" im UI. Prüfen Sie nun in Ihrem Browser das Remote Repository auf GitHub.com – dort sollte in Ihrem Repository das neue Unterverzeichnis "`HelloGit`" zu sehen sein.
Machen Sie sich nachfolgend auch mit den Kommandos `fetch` und `pull` vertraut (Kommandozeile bzw. im UI).

2) In dieser Aufgabe üben Sie das gemeinsame Arbeiten an einem Projekt. Wir verwenden hierzu das Repository "`TestRepo`", welches Ihr erstelltes Unterverzeichnis "`HelloGit`" enhält (Aufgabe 1). Die dort befindlichen Dateien "`Hello.c`" und "`Greeter.c`" wollen wir für diese Übung nutzen. Sprechen Sie sich untereinander ab, wer von beiden sein Repository hierfür zur Verfügung stellt (Sie können auch beide Varianten durchführen).

   Im Folgenden gilt: Student S1 stellt das Repository zur Verfügung; Student S2 trägt als Kollaborator bei.

   Die Schritte führen Sie idealerweise aus, wenn Sie beide per Chat oder Audio verbunden sind. Die Beschreibung setzt voraus, dass Sie GitHub Desktop nutzen. Es kann jedoch auch ein anderer Client oder die Kommandozeile benutzt werden (siehe Zusatzaufgabe).

   S1: Fügen Sie S2 als Kollaborator zu Ihrem Repository hinzu: Öffnen Sie Ihr Repository "`TestRepo`" im Browser, wählen Sie "Settings" und dann Links im Menü "Manage Access -> Collaborators". Hier suchen Sie S2 anhand der Kennung und fügen diese hinzu.

   S2: Klonen Sie das Repository in ein lokales Verzeichnis. Achtung: Sie haben schon ein Verzeichnis "`TestRepo`" in Ihrem "GitHub"-Verzeichnis. Legen Sie das neue Repository daher am besten in einem Unterverzeichnis ab, welches als Name z.B. die Kennung von S1 trägt.

   S1: Fügen Sie einen Kopf-Kommentar zur Funktion `main()` in `Hello.c` hinzu.

   S1: Führen Sie einen Commit aus und übertragen die Änderungen an den Server (Push).

   S2: Fügen Sie einen Kommentar innerhalb der Funktion `main()` in `Hello.c` hinzu.

   S2: Führen Sie einen Commit aus und übertragen Sie die Änderungen an den Server (Push). Was passiert beim letzten Schritt (Push) von S2? Verfolgen Sie die Änderungen im "History"-Tab von Github Desktop nach.

   S1: Führen Sie einen Pull aus und verfolgen die Änderungen ebenfalls über den History-Tab nach.

S1: Ändern Sie die Datei `Greeter.c`, indem Sie das Wort "`wonderful`" durch "`snowy`" ersetzen. Erstellen Sie einen Commit und führen einen Push auf den Server durch.

S2: Ändern Sie ebenfalls die Datei `Greeter.c`, indem Sie das Wort "`wonderful`" durch "`cloudy`" ersetzen. Erstellen Sie einen Commit und führen einen Push auf den Server durch. Ihnen sollte nun ein Konflikt angezeigt werden. Das Dialogfenster bietet an, diesen im Editor zu bearbeiten. Tun Sie dies, indem Sie schlussendlich "`sunny`" als Kompromiss verwenden. 😊 Nach dem Speichern der Datei erlaubt Ihnen GitHub Desktop, einen neuen Commit per "Commit Merge" zu erstellen. Verfolgen die Änderungen nochmals über den History-Tab nach und führen dann einen Push auf den Server durch.

S1: Führen Sie einen Pull aus und verfolgen die Änderungen ebenfalls über den History-Tab in GitHub Desktop nach.

*Zusatzaufgabe für Fortgeschrittene: Verwenden Sie statt GitHub Desktop die Kommandozeile, um diese Operationen durchzuführen.