Allgemeine Hinweise zu den SP-Übungen

- Die Aufgaben sind teils alleine, teils in Zweiergruppen zu bearbeiten (siehe Aufgabenstellung). Bei Gruppenarbeit sind Lösungsweg und Programmierung gemeinsam zu erarbeiten.
- Zur Versionsverwaltung wird `git` eingesetzt. Dafür muss ein an das IDM gekoppelter Account auf `https://gitlab.cs.fau.de` erstellt werden.
- Für jede Aufgabe ist die Bearbeitungszeit in Werktagen (bei uns nur Montag bis Freitag) angegeben. Die Bearbeitungszeit beinhaltet den Tag der eigenen Tafelübung. Nicht enthalten sind vorlesungsfreie Tage und Feiertage. Mit Hilfe des Kommandos `/proj/i4sp1/bin/get-deadline <aufgabeX>` kann der effektive Abgabetermin für den Aufrufenden angezeigt werden.
- Die Übungsaufgaben müssen spätestens bis zum jeweiligen Abgabetermin abgegeben werden. In darauffolgenden Tafelübungen werden einzelne abgegebene Lösungen besprochen – jeder Übungsteilnehmer muss dabei in der Lage sein, die gesamte Lösung seiner Gruppe zu erläutern. Kann jemand seine Lösung auf Anforderung nicht erklären, wird für ihn die Aufgabe als nicht abgegeben bewertet (im Zweifelsfall kann hierzu ein Gespräch außerhalb der Tafelübung stattfinden).
- Die abgegebenen Programme werden automatisch auf Ähnlichkeit mit anderen Programmen desselben Semesters und früherer Semester überprüft. Werden starke Übereinstimmungen festgestellt, wird die Aufgabe als nicht abgegeben bewertet.
- Kann ein Übungstermin nicht wahrgenommen werden, kann vorher mit dem eigenen Übungsleiter ein Ersatztermin in einer anderen Übungsgruppe vereinbart werden.
- Durch ausführen des Skriptes `/proj/i4sp1/bin/mkrepo` kann für jede Aufgabe ein neues Projekt-Repository im Gitlab erzeugt werden. Dieses wird automatisch mit den Vorgabedateien befüllt.
- Der letzte Stand (am Hauptzweig **main**), der zum Abgabezeitpunkt in dem Gitlab-Repository (`git push`) vorgefunden wird, wird als Abgabe gewertet.
- Sollten Sie aus triftigem Grund eine fristgerechte Abgabe versäumen, so ist eine verspätete Abgabe durch Aufruf des Abgabeprogramms möglich. Damit eine solche Abgabe bei der Wertung berücksichtigt wird, ist jedoch in jedem Fall eine Rücksprache mit Ihrem Übungsleiter erforderlich. Eine vorhandene rechtzeitige Abgabe wird durch eine verspätete Abgabe nicht gelöscht.

Wichtig: Lesen Sie auch den Teil "**Hinweise zur Aufgabe**" auf diesem Blatt; Spezifikationen in diesem Teil sind ebenfalls einzuhalten!

Aufgabe 1: lilo (4.0 Punkte)

Implementieren Sie eine einfach verkettete Liste, welche nicht-negative Ganzzahlen verwaltet. Auf die Liste soll mit den folgenden Funktionen zugegriffen werden:

- `int insertElement(int value)`: Fügt einen Wert in die Liste ein, wenn dieser noch nicht vorhanden ist. Im Erfolgsfall gibt die Funktion den eingefügten Wert zurück, ansonsten den Wert -1.
- `int removeElement(void)`: Entnimmt den ältesten Wert aus der Liste und gibt diesen zurück. Ist kein Wert in der Liste vorhanden, wird -1 zurückgeliefert.

Erstellen Sie sich mithilfe des Skriptes `/proj/i4sp1/bin/mkrepo` ein Git-Repository. Hiermit erhalten Sie eine Vorlage für die Quelldatei. Das darin enthaltende Hauptprogramm (Funktion `main()`) fügt einige Werte in die Liste ein und entnimmt diese wieder. Die Codesequenz aus der Vorlage soll folgende Ausgabe erzeugen:

```
insert 47: 47
insert 11: 11
insert 23: 23
insert 11: -1
remove: 47
remove: 11
```

Hinweise zur Aufgabe:

- Erforderliche Dateien: `lilo.c` (4 Punkte)
- Hilfreiche *Manual-Pages*: **malloc(3)**, **free(3)**
- Das C-Programm ist in der Datei `lilo.c` abzulegen. Es muss dem ANSI-C11-Standard entsprechen und mit dem GNU-C-Compiler auf den Linux-Rechnern im CIP-Pool kompilieren. Dazu ist der Compiler mit folgenden Parametern aufzurufen.
  `gcc -std=c11 -pedantic -D_XOPEN_SOURCE=700 -Wall -Werror -o lilo lilo.c`
- Implementieren Sie keine Listenfunktionalität in der Funktion `main()`. Allerdings können Sie die Funktion `main()` erweitern, um Ihre Implementierung zu testen.
- Der Versuch, eine negative Zahl in die Liste einzufügen, soll unterbunden und als Fehler gewertet werden.
- Sollte bei der Ausführung einer verwendeten Funktion (z.B. `malloc(3)`) ein Fehler auftreten, sind keine Fehlermeldungen auszugeben – der Fehler muss aber gemäß der Aufgabenstellung sinnvoll behandelt werden.
- Unterprogramme und globale Variablendefinitionen sind ausreichend zu kommentieren. Achten Sie bitte außerdem auf saubere Gliederung des Quellcodes!

Hinweise zur Abgabe:

Bearbeitung: Einzeln
Bearbeitungszeit: 6 Werktage (ohne Wochenenden und Feiertage)
Abgabezeit: 17:30 Uhr