Universität Konstanz
Informatik & Informationswissenschaft
Prof. Dr. Marcel Waldvogel / Klaus Herberth

Programmierkurs 3
WS 2016/2017

8. Übungsblatt

(Ausgabe: 15. Dezember 2016 — Abgabe bis: 22. Dezember 2016, 4:00)

Aufgabe 1: Reverse Polish Notation (17 Punkte)

Schreiben Sie einen Rechner (`calc`), der als Kommandozeilenargumente Rechenaufgaben in RPN mit Ganzzahlen erwartet und das korrekte Ergebnis auf der Standardausgabe ausgibt. Ihr Rechner sollte dabei alle 4 Grundrechenarten beherrschen. Achten Sie darauf, dass etwaige Falscheingaben entsprechend behandelt werden. Nutzen Sie als Stack die bereitgestellte verlinkte Liste die ihnen vom letzten Übungsblatt in veränderter Form bekannt ist und binden Sie sie in ihren Quelltext in der Datei `calc.c` ein. Eine Beispielausführung würde wie folgt aussehen:

```
$ ./calc 5 4+ 2 *
18
```

Aufgabe 2: Preprocessor (2 Punkte)

Fügen Sie `#define FOO` so ans Ende der Datei `list.h` ein, dass es weder ein Kommentar ist, noch dass das restliche Dokument verändert werden muss, noch dass es eine Compiler Warnung gibt.

Aufgabe 3: Debugging (4 Punkte)

Ändern Sie ihr Programm aus Aufgabe 1 so, dass es wenn es mit folgenden Flags kompiliert wird, während der Laufzeit Informationen zu seinem aktuellen Status ausgibt.

```
$ pkgcc -D DEBUG ...
$ ./calc
Lese Zahl
Lese Zahl
Lese Operator
Berechne Zwischenergebnis
...
```

`pkgcc` beschreibt hierbei `gcc` mit allen zu verwendeten Flags, wie in der Vorlesung gezeigt.

Aufgabe 4: Fehlerbeschreibung (2 Punkte)

Sollte ihr Programm aus Aufgabe 1 einen Fehler melden (z.B. weil die Eingabe ungültig ist) geben Sie mit `printf` den Grund, die aktuelle Zeilennummer und den Dateinamen aus. Es sollte auch korrekt funktionieren, wenn oberhalb der Zeile Code geändert bzw. die Datei umbenannt wird.

Aufgabe 5: Makefiles **(10 Punkte)**

Schreiben Sie ein GNU-Makefile, dass Ihnen beim Kompilieren von Aufgabe 1 helfen soll. Schreiben Sie das Makefile in der Form, dass nur aktualisierte Dateien neu kompiliert werden. Halten Sie sich dabei an die GNU Coding Standards. Fügen Sie außerdem ein Target `clean` ein, dass das aktuelle Verzeichnis wieder aufräumt und ein Target namens `precommit` welches checkpatch auf ihren Dateien ausführt und ihnen Rückmeldung darüber gibt, ob ihre Namen in allen notwendigen Dokumenten sind.

Aufgabe 6: Advanced Makefile **(10 Punkte)**

Erklären Sie wie das auf Folie 212 gezeigte GNU Makefile funktioniert und gehen Sie besonders auf die rot markierten Passagen ein.

Aufgabe 7: Testing **(5 Punkte)**

Fügen Sie der bereitgestellten Verketteten Liste eine Funktion `rem` hinzu die das entsprechende Element aus der Liste löscht. Erstellen Sie zudem eine Datei `testList.c` die alle Funktionen der Liste testet und möglichst viele potentielle Fehlerfälle abdeckt. Korrigieren Sie mögliche Fehler. Der Test sollte mit `make check` starten.

Aufgabe 8: Dokumentation **(10 Punkte)**

Nutzen Sie das Programm `doxygen` um HTML Dokumentation von all ihren auf diesem Übungsblatt erstellten Dateien und Funktionen zu generieren. Erstellen Sie auch hierfür ein, nach den GNU Coding Standards korrekten, Target in ihrem Makefile.