Wichtige Hinweise zur Anfertigung von Lösungen der Übungsaufgaben

Stefan Kurtz
Arbeitsgruppe Genominformatik,
Zentrum für Bioinformatik Hamburg

19. Oktober 2021

Damit die Übungsleiter bzw. Korrekturtutoren sich beim Korrigieren der Lösungen auf die Inhalte konzentrieren können, müssen die Lösungen die folgenden Bedingungen erfüllen:

1 Regeln zur Annahme der Lösungen

1. Die Lösungen müssen jeweils bis zum im Übungsblatt angegebenen Zeitpunkt an die auf dem Übungsblatt angegebene E-mail Adresse geschickt werden, also

   modul`@zbh.uni-hamburg.de`

   wobei *modul* die Abkürzung für das Modul in Kleinschreibung ist, also entweder `pfn1`, `gsa`, `pfn2` oder `gik`.
2. Bitte erstellen Sie Ihre Lösungen in Gruppenarbeit. Jede Gruppe sollte 2-3 Studierende umfassen.

2 Regeln zum Format der Abgabe

1. Das Material zu einem Übungsblatt liegt in einer Verzeichnisstruktur vor, die unbedingt in der Abgabe Ihrer Lösung erhalten bleiben muss. Im Übungsblatt sind in fast allen Fällen Dateinamen genannt, die Sie für Ihre Lösung verwenden müssen.
2. Die genannte Verzeichnisstruktur (Details siehe unten) muss in einem gzip-tar-Archiv (d.h. einer `.tar.gz`-Datei) verpackt werden und dieses muss ein Anhang Ihrer E-mail sein. Die E-mail und die Betreff-Zeile sollten keine weitere Information enthalten, denn wir extrahieren mit einem Programm die `.tar.gz` Datei aus der E-mail und lesen die E-mail und das Subject selbst nicht.

3. Falls Ihre Lösung eine PDF-Datei oder Dateien in Formaten von Textverarbeitungsprogrammen (z.B. mit einem Beweis) enthält, dann bitte die Zeilen nummerieren, damit bei der Korrektur auf die entsprechenden Stellen verwiesen werden kann. Falls Sie LaTeX verwenden, schreiben Sie

   ```
   \usepackage{lineno}
   \linenumbers
   ```

   in die Präambel (also vor `\begin{document}`) Ihres Dokuments, um die Zeilennummern auszugeben.

4. Der Dateiname des Archivs muss dem folgenden Schema entsprechen:

   `Blatt05.Nn1.Nn2[...].tar.gz`

   Dabei sind `Nn1` und `Nn2` die Nachnamen der Gruppenmitglieder, in alphabetischer Reihenfolge. Die Nachnamen beginnen jeweils mit einem Großbuchstaben und sind ansonsten klein geschrieben. Ein solches Archiv erzeugen Sie durch

   `tar -cvzf Blatt05.Nn1.Nn2[...].tar.gz Blatt05.Nn1.Nn2[...]`

   Natürlich ändert sich im Laufe des Semesters die Nummer des Blattes.

   Wir haben in früheren Übungen manchmal Abgaben erhalten, bei denen die Namen im tar-archiv oder im Verzeichnis vergessen wurden, so dass z.B. eine Datei mit dem Namen `Blatt05.tar.gz` angehängt wurde oder das Verzeichnis nur `Blatt05` hieß. In einem solchen Fall können wir die Lösungen nicht mehr automatisch einer Gruppe zuordnen. Wenn zudem eine solche Abgabe mehrfach auftritt, dann gehen Lösungen verloren, weil nach dem Auspacken der `.tar.gz` Dateien Verzeichnisse mit dem gleichen Namen überschrieben werden. Eine manuelle Nachkorrektur ist bei der Vielzahl der Abgaben sehr zeitaufwändig. **Daher müssen Sie die Namenskonventionen unbedingt einhalten.**

5. Um das Erstellen der `.tar.gz`-Datei zu vereinfachen, gibt es im Repository mit den Materialien unter `bin/tar_loesungen.py` ein Skript. Zur Verwendung des Skriptes müssen Sie lediglich eine Datei `namen.txt` anlegen, die zeilenweise in lexikographischer Reihenfolge die Nachnamen der Mitglieder Ihrer Gruppe enthält. Diese Datei muss nur einmal angelegt werden. Ein Änderung ist nur erforderlich, wenn sich die Gruppenzusammensetzung ändert.

6. Der Name des Archivs und die Namen aller darin enthaltenen Dateien dürfen keine Leerzeichen, Umlaute oder ß enthalten. Umlaute in Ihren Namen müssen durch ae, oe, ue, und ss ersetzt werden und bei Nachnamen, die aus mehreren Worten bestehen, müssen diese durch einen Unterstrich _ voneinander getrennt werden. Bei einem Namen, der einen Bindestrich enthält, bleibt dieser erhalten. Diese Regeln sind so genau definiert und müssen eingehalten werden, damit Ihre Abgaben automatisch E-mail Adressen zugeordnet werden können, an die wir die korrigierten Lösungen zurückschicken. Falls die Regeln spezifische Eigenschaften Ihres Nachnamens noch nicht berücksichtigen, informieren Sie bitte Herrn Kurtz.

7. Das Archiv darf keine Binärdateien (z.B. Objektdateien oder kompilierte Programme) enthalten; dies führt nämlich ggf. dazu, dass die Email ohne Rückmeldung nicht zugestellt wird (Spamfilter). Sie müssen daher vor der Abgabe in jedem Unterverzeichnis, dass eine Datei mit dem Namen `Makefile` enthält, einmal `make clean` aufrufen.
8. Es kommt gelegentlich vor, dass die abgegebenen Verzeichnisse Dateien enthalten, die durch die Verwendung anderer Programme (z.B. IDEs oder Textverarbeitungssystemen) entstanden sind. Typisch sind hier Dateien mit dem Namen `.DS_Store` oder `__pycache__`. Solche Dateien sollen aus dem Verzeichnisbaum gelöscht werden, bevor Sie die tar-Dateien erstellen.

Hier ist ein Beispiel für die Verzeichnisstruktur von Blatt05 für die Gruppe mit den Mitgliedern *Hilbert* und *Mößner* im Modul PfN1:

```
--Blatt05.Hilbert.Moessner
  |
  |--Aufgabe1
  |  |-- bearbeitung.txt
  |  |-- HelloSimple
         |-- Makefile
             hello.py
             ...
  |--Aufgabe2
  |  |-- bearbeitung.txt
  |  |-- LimitsTsv
         |-- Makefile
             limits_tsv.py
```

Name der entstehenden `.tar.gz`-Datei: `Blatt05.Hilbert.Moessner.tar.gz`. Diese wird als E-mail Anhang an `pfn1@zbh.uni-hamburg.de` geschickt.

Bei Missachtung einer dieser Regeln wird (ab Übungsblatt 3) maximal 0.5 Punkt pro Übungsblatt abgezogen.

3 Regeln zu den Testdaten und Testskripten

1. Das Material enthält Dateien zum Testen Ihrer Lösungen. Diese dürfen Sie nicht löschen und auch nicht ändern, es sei denn, es ist in der Aufgabenstellung explizit so vermerkt.

4 Regeln zur Kompilierbarkeit und Ausführbarkeit von Programmcode

1. Der Programmcode muss kompilierbar sein (bei C/C++) bzw. ausführbar sein (bei Skriptsprachen). Nur Skripte selbst dürfen ausführbar sein, wenn sie ein Hauptprogramm ent-

halten. Ein Makefile oder C/C++ Sourcen und andere Textdateien dürfen nicht ausführbar sein.

Bei Missachtung einer dieser Regeln wird (ab Übungsblatt 3) maximal 1 Punkt pro Aufgabe abgezogen.

5 Regeln zum Stil und Format der Quellcode-Dateien

1. Die Aufgabenstellungen geben jeweils an, wie die Dateien heißen. Diese Dateinamen müssen Sie verwenden, weil sonst die Tests nicht funktionieren.
2. Falls Ihr Programmcode nicht lauffähig ist oder die Tests nicht besteht, müssen Sie dieses unbedingt dokumentieren und einen Hinweis geben, wo die Probleme Ihrer Ansicht nach liegen. Diese Dokumentation muss am Anfang der entsprechenden Dateien stehen, damit sie nicht übersehen wird.
3. für PfN2: C/C++: Variablen dürfen nur am Anfang eines Blocks deklariert werden (C89 Standard). Das verbessert die Lesbarkeit des Codes.
4. Es dürfen nur die Skripte im Verzeichnis ein X-bit haben (d.h. ausführbar sein). Alle anderen Dateien sind nicht ausführbar. Für MS-Windows Nutzer: Da nach Bearbeitung mit MS-Windows die ursprünglichen Rechte der Dateien verändert werden, müssen Sie diese selbst wieder vor Abgabe mit `chmod -x *.py Makefile` etc zurücksetzen.
5. Die Einrückung muss korrekt sein: hierbei konsistent 2 oder 4 Leerzeichen, aber keine Tabulatoren verwenden (außer im Makefile, in dem Tabulatoren notwendig sind). Generell dürfen Tabulatoren in Dateien mit Programmcode nicht verwendet werden. In viele Editoren lassen sich entsprechende Einstellungen konfigurieren.
6. Zeilenumbrüche müssen im Unix-Format sein (das ist in vielen Editoren unter Windows und Mac auch einstellbar). Die Konvertierung können Sie durch das Programm `dos2unix` vornehmen, das Sie ggf. nachinstallieren müssen. Manche E-mail Clients konvertieren automatisch Zeilenumbrüche in Textdateien, die als Anhang verschickt werden. Sie sollten sich dieser Problematik bewusst sein, und ggf. innerhalb Ihrer Gruppe die Dateien als `.tar.gz` austauschen oder `git` verwenden.
7. Die Länge einer Zeile darf höchstens 80 Zeichen sein, da längere Zeilen nicht gut lesbar sind. Diese Regel gilt auch für Textdateien, die keinen Programmcode, sondern z.B. Beweise enthalten. Sie können mit `fold -w 80 -s inputfile` eine Eingabedatei in Zeilen der angegebenen Länge splitten. Für Python: lange Zeilen sollen an passenden Positionen umgebrochen werden oder das `\`-Zeichen unmittelbar vor dem Zeilenumbruch verwendet werden.
8. Im Verzeichnis `bin/code_check.py` finden Sie ein Skript, das die Zeilenlänge und (optional für Python-Programme) die Einrückung überprüft und entsprechende Fehlermeldungen ausgibt, falls die Regeln nicht eingehalten werden.
9. Für PfN2: Im git repo mit den Materialien finden Sie unter `doc/clang-format.txt`

Hinweise zur Formatierung von C-Programmen mit dem Programm `clang-format`. Wenn Sie `clang-format` so verwenden, wie dort beschrieben, dann werden Ihre Programme sinnvoll formatiert. Insbesondere wird die Breite einer Zeile auf 80 Zeichen begrenzt. Code-Style Anforderungen, wie die ausschließliche Deklaration von Variablen am Anfang eines Blocks kann Ihnen clang-format selbstverständlich nicht abnehmen.

Bei Missachtung einer dieser Regeln wird (ab Übungsblatt 3) maximal 1 Punkt pro Aufgabe abgezogen.

6 Regeln zur korrekten Lösung

1. Wenn Tests vorhanden sind, müssen die Programme diese bestehen. Diese sind i.d.R. durch den Befehl `make test` ausführbar.
2. Die Test-Dateien und Test-Ziele dürfen nicht verändert werden, um dadurch die Tests zu bestehen.
3. Viele der gestellten Aufgaben enthalten Hinweise zu den zu entwickelnden Programmteilen einschließlich der entsprechenden Schnittstellen. Die Hinweise orientieren sich an der Musterlösung und sollen Ihnen helfen, mit vertretbarem Aufwand zu einer Lösung zu kommen, die dann auch die Tests besteht. Wenn Sie sich zutrauen, eine Lösung mit einer anderen als der vorgegebenen Struktur zu entwickeln, können Sie das gerne tun.

Bei Missachtung einer dieser Regeln wird (ab Übungsblatt 3) maximal 1 Punkt pro Aufgabe abgezogen.

7 Regeln zur Angabe der Metainformation

1. Im Verzeichnis jeder Aufgabe finden Sie eine Datei mit dem Namen `bearbeitung.txt`. Hier tragen Sie bitte für den Platzhalter *h* im folgenden Format die Anzahl der Stunden ein, die sie für die Aufgabe aufgewendet haben.

   `Bearbeitungszeit:` *h* `Stunden`

   Das können auch Fließpunkt-Werte sein, also z.B. 1.75, wenn Sie 1 Stunde und 45 Minuten benötigt haben. In einer weiteren Zeile tragen Sie im folgenden Format für den Platzhalter *g* ein, wie schwierig Sie die Aufgabe fanden.

   `Schwierigkeitsgrad:` *g*

   Dabei sind folgende Werte möglich:

   `leicht, mittelschwer, genau richtig, schwer, sehr schwer`

   In einer weiteren Zeile tragen Sie im Format für den Platzhalter *a* ein, wie verständlich die Aufgabenstellung formuliert war:

   `Aufgabenstellung:` *a*

Dabei sind folgende Werte möglich:

`vollkommen klar, weitgehend klar, weniger klar, unklar`

Falls eine der beiden letzten Kategorien zutrifft, bitte kurz angeben, worin die Unklarheiten bestehen.

Diese Angaben erlauben uns systematisch ihren Aufwand und Ihren Eindruck von den Aufgaben zu erfassen und ggf. die Aufgabenstellung für die Zukunft anzupassen. Da diese Zeilen von einem Skript verarbeitet werden (um die Daten statistisch auszuwerten), ist es wichtig, dass Sie dieses Format einhalten.

8 Regeln zur guten Programmierpraxis

1. Der Rückgabewert der Standard-Library Funktionen (z.B. `fopen`, `malloc` in C) muss immer überprüft werden.
2. Das Programm muss robust sein: z.B. bei falschen Eingaben muss immer eine sinnvolle Fehlermeldung auf den Standard-Fehlerkanal geschrieben werden, und das Programm muss **mit einem Exit-Code** (z.B. `EXIT_FAILURE` bei C/C++-Programmen oder `exit(1)` in Python-Skripten) beendet werden.
3. Wird das Programm mit falschen Argumenten aufgerufen, muss die korrekte Syntax in der Fehlermeldung in Form einer `Usage:`-Zeile angegebenen werden. Eine solche Zeile enthält immer den Pfad des ausgeführten Programms (steht in `argv[0]` in C oder `sys.argv[0]` in Python) und die erwarteten Argumente. Beispiel:

   `Usage: ./feature_eval.x <inputfile>`

Bei Missachtung einer dieser Regeln wird (ab Übungsblatt 3) maximal 1 Punkt pro Aufgabe abgezogen.

9 Checkliste vor Abgabe der Lösungen

Hier finden Sie noch einmal die wichtigsten Dinge, die Sie beachten müssen, bevor Sie Ihre Lösungen abgeben.

1. Wechseln Sie in das Verzeichnis mit Ihren Lösungen.
2. Benennen Sie ggf. Ihr Verzeichnis um. Im Material heißt es z.B. `Blatt01`. Ihr Verzeichnisname muss nach `Blatt01.` die Nachnamen der Gruppenmitglieder, jeweils durch einen Punkt getrennt enthalten.
3. Verifizieren Sie, dass die Tests funktionieren. Falls nicht, müssen entsprechende Kommentare eingefügt werden.
4. Rufen Sie in den Verzeichnissen mit einem `Makefile` den Befehl `make clean` auf.

5. Verifizieren Sie, dass Sie für alle Aufgaben die Angaben in der entsprechenden Datei `bearbeitung.txt` gemacht haben.
6. Überprüfen Sie mit `check_code.py` die Formatierung Ihrer Dateien.
7. Rufen Sie `tar_loesungen.py N` auf, wobei `N` die Nummer des Übungblattes ist.
8. Wenn alles in Ordnung ist, dann wurde eine `.tar.gz`-Datei mit den Namen erzeugt, die sich aus der Blattnummer und den Namen in der Datei `namen.txt` ergibt.
9. Die `.tar.gz`-Datei schicken Sie als Anhang an die E-mail Adresse, die mit der Abkürzung des Modulnamens in Kleinschreibweise beginnt.