- Nachfolgend finden Sie zwei weitere Beispielaufgaben zum Ausprobieren der Onlinetest-Umgebung. Sie können weiterhin die Zugangsinformationen für die Testumgebung nutzen, die Sie beim Herunterladen des ersten Übungsblatts über `https://www.mi.hs-rm.de/~weitz/cgi-bin/aufgabenblatt.cgi` erhalten haben.

- Die **Vorlesungsfolien** und **Python-Bibliotheksreferenz** können über das Menü "... Skript" bzw. "Python Library" als PDF eingesehen werden, das Online-Manual mit dem Kommando `man`. Weitere **Hilfsmittel** sind **nicht** zugelassen.

- **Eigener Testcode** kann in der vorgegebenen `main()`-Funktion angelegt werden, Sie können Ihren Code von dort aus wie gewohnt ausführen und debuggen. Ihre Lösung darf allerdings **weder** bei `import` Ihres Moduls (Python) **noch** bei der Ausführung der Lösungs-Bestandteile **eigene (nicht verlangte) Ausgaben** machen (in Ihrer `main()` können Sie beliebige Hilfs-/Testausgaben unterbringen). Verlangte Ausgaben müssen dem vorgegebenen Format entsprechen.

- Mit dem `pruef`-Kommando (z.B. `pruef a1.py`) können Sie Ihre Lösung automatisiert und **unverbindlich vor-testen** lassen. Beachten Sie bitte die ausgegebenen Meldungen. Mit `pruef log` können Sie sich das letzte Prüfprotokoll noch einmal ausgeben lassen. Das Prüfprogramm legt Backups halbwegs erfolgreicher getesteter Programme im Verzeichnis `backup.pruef` ab. Sie können während des Tests darauf zurückgreifen, **gewertet** werden aber **nur** die **vorgabegemäß benannten** Dateien im **vorgegebenen Verzeichnis**. Legen Sie daher bitte keine eigenen (Unter-)Verzeichnisse an. Lassen Sie vor dem Ausloggen nochmal ein `pruef` über Ihre Lösugen laufen, um sicher zu stellen, dass Sie nicht aus Versehen noch etwas verändert/beschädigt haben. Um das `pruef`-Programm sinnvoll nutzen zu können, müssen die in der Aufgabenstellung geforderten Funktionen/Klassen/Typen o.ä. **zumindest angelegt** sein (ggf. zunächst einfach "leer").

- Die Lösungen müssen in ANSI-C/Python 3 geschrieben, fehler- und warnungsfrei sein **und** ohne Laufzeitfehler die erwarteten Ergebnisse in der vorgegebenen Form liefern. Ihre **Lösungen** müssen **gemäß Aufgabenstellung allgemein**, also auch mit variierten Tests, funktionieren – beispielsweise gilt die Erzeugung von Ergebnissen anhand fester Test-Ein-/Ausgabewertkombinationen als Täuschungsversuch. Bitte betrachten Sie die Tests mit dem `pruef`-Programm auch als Illustration/Beispiel zur Aufgabe.

- C: Legen Sie Ihre C-Lösungen bitte in Ihrem Login-Verzeichnis unter Einhaltung der vorgegebenen Namen an (keine eigenen Unterverzeichnisse o.ä.). Zum Starten/Debuggen verwenden Sie *nur* den Namen des ausführbaren Programms (z.B. `a.out` oder `a17`), also *ohne* führendes "./" wie in den Praktika. Die Lösung muss ohne `valgrind`-Beanstandungen wie Speicherlecks oder unzulässige Speicherzugriffe durchlaufen.

  Ihnen stehen in der **Toolbar** (untere Bildschirmkante) Buttons zum Aufruf von Geany, Nemiver und eines Terminals zur Verfügung, das in Ihrem Homedirectory startet. Sie finden Ihre C-Dateien direkt dort vor.

- Python: Nach Start von **Eclipse** sehen Sie ein **vorbereitetes Python-Projekt** `eclipse`. Legen Sie Ihre (ANSI-C/Python 3-)Lösungen bitte direkt dort unter Einhaltung der vorgegebenen Namen an (keine eigenen Unterverzeichnisse/Pakete o.ä.). Bitte führen Sie zu Beginn mit dem Rechte-Maustaste-Menü ein "**Refresh**" auf dem Projekt `eclipse` aus.

  Ihnen stehen in der **Toolbar** (untere Bildschirmkante) Buttons zum Aufruf von Eclipse und eines Terminals zur Verfügung, das im vorbereiteten Eclipse-Projektverzeichnis startet, Sie finden Ihre Python-Dateien direkt dort vor.

Aufgabe 3 (8 Punkte)

Das hawaiianische Alphabet kennt nur die Vokale A, E, I, O, U sowie die Konsonanten H, K, L, M, N, P, W (jeweils in Groß- und Kleinschrift). Der Apostroph (') wird als kurze Pause ausgesprochen und wie ein weiterer Konsonant behandelt.

Für hawaiianische Wörter gelten folgende Regeln:

- Alle Wörter sind (nur) aus dem hawaiianischen Alphabet zusammengesetzt.
- Alle Wörter enden mit einem Vokal,
- auf einen Konsonanten folgt stets ein Vokal und
- Silben enden nie mit Konsonanten. Der Einfachheit halber nehmen wir an, dass innerhalb eines Wortes **nur** bei einem Apostroph eine rechtschreibrelevante neue Silbe beginnt, andere Silbengrenzen ignorieren wir.

4 Pkt Bitte schreiben Sie eine Funktion `isHawaiian(w)`, welche für den übergebenen String `w` überprüft, ob er ein hawaiianisches Wort ist, und den entsprechenden `bool`-Wert für *wahr* zurückgibt, falls dies der Fall ist, ansonsten den für *falsch*.

Beispiele für hawaiianische Wörter wären also `Aloha`, `Hawai'i`, `Wahine` *(Frau)*, `wikiwiki` *(schnell)* und `Humuhumunukunukuapua'a` *(der offizielle Staatsfisch)*.

Nicht-hawaiianische Wörter sind dagegen (Fehlerstellen unterstrichen) z.B. `Qui'juno`, `Alaaf`, `lekker`, `Pu'kuuli` und `ohamak'uluu`.

4 Pkt Nehmen wir an, wir hätten ein beliebiges **iterierbares Objekt** `iterable`, das eine (beliebig lange) **Folge von Zeichenketten** (nicht unbedingt nur einzelne Wörter) liefert.

Jede Zeichenkette besteht aus Leerzeichen-getrennten Wörtern, ggf. mit Satzzeichen. Wenn Satzzeichen (Punkt, Komma, Doppelpunkt, Ausrufungs- und Fragezeichen) vorhanden sind, folgen sie immer unmittelbar auf ein Wort, treten also nicht "irgendwo" unerwartet auf. Satzzeichen werden aber natürlich nicht als Teil eines Wortes betrachtet. **Beispiel**: Die Zeichenkette "`Hallo, Du: Wo ist der Bahnhof?`" besteht in diesem Sinne aus den sechs Wörtern `Hallo` `Du` `Wo` `ist` `der` `Bahnhof`

Schreiben Sie nun bitte eine **Generator-Funktion** `extractHi(iterable)`, welche ein solches, Zeichenketten-lieferndes `iterable` entgegennimmt und ein **Generatorobjekt zurückgibt**, welches der Reihe nach alle hawaiianischen Wörter aus dem übergebenen `iterable` liefert (und sonst nichts, Reihenfolge wie in `iterable`).

Beispiele:

- `extractHi( ["Eine Wahine sagt", "Maika'i no au!"] )`
  liefert ein Generatorobjekt, das diese Stringfolge erzeugt:
  "Eine", "Wahine", "Maika'i", "no", "au"
- `extractHi( ("Hau'oli", "la hanau!", "Happy","Birthday!") )`
  liefert ein Generatorobjekt, das diese Stringfolge erzeugt:
  "Hau'oli", "la", "hanau".
- `extractHi( open("hawaii-datei.txt") )`
  liefert ein Generatorobjekt, das Stringfolge der hawaiianischen Wörter aus dem übergebenen Dateiobjekt erzeugt.

Aufgabe 4 (8 Punkte)

Die Völkerverständigung ist eine wichtige Sache. Schreiben Sie bitte eine C-Datei `a4.c`, die eine Funktion

void schwabify(char *input, char output[])

bereitstellt.

Die Funktion verarbeitet einen Eingabestring `input`, indem sie ihn näherungsweise in das Schwäbische übersetzt. Das Ergebnis wird (ebenfalls als C-String) in `output` geschrieben, der genügend Platz für das Ergebnis hat, jedoch **nicht** vorab initialisiert ist (enthält also bei Aufruf eine zufällige Bytefolge).

Leerzeichen bleiben erhalten. Der Eingabestring `input` wird nicht verändert. Der Aufrufer ist dafür verantwortlich, dass `output` groß genug ist, um den übersetzten `input` zu fassen (ist also nicht Ihr Problem).

Folgende Umformungen nimmt `schwabify()` dabei vor:

- Ein Punkt bedeutet Satzende, er wird durch die Wendung "`, woisch?`" ersetzt.
- Jedes Auftreten der Zeichenkette "`halt`" wird durch "`heb`" ersetzt.
- Jedes Auftreten der Zeichenkette "`ist`" wird durch "`isch`" ersetzt.

Sie brauchen nur die Wörter in der gezeigten (Klein-)Schreibung zu behandeln.

Beispiele:

```
  schwabify("Das ist interessant.", output);
/* output = "Das isch interessant, woisch?" /*

  schwabify("Mit     diesem Kleber     sollte es halten, oder?", output);
/* output = "Mit     diesem Kleber     sollte es heben, oder?" */

  schwabify("haltisthalt. isthaltist...", output);
/* output = "hebischheb, woisch? ischhebisch, woisch?, woisch?, woisch?" */
```

Wenn Sie möchten, können Sie auf die C-Stringfunktionen aus der Standardbibliothek (`man string`) zurückgreifen, soweit sinnvoll.