FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Algorithmen & Datenstrukturen ILV
Hashtabellen

Abgabe: Details zu den Abgabemodalitäten finden Sie auf Moodle. Voraussetzung für die Bewertung ist, dass das Programm kompiliert und ausführbar ist. Die Lösungen werden jeweils am Anfang der nächsten (online) Übungseinheit von zufällig gewählten Studierenden vorgestellt.

Ziele:

- Funktionsweise und Kennenlernen von Hashtabellen

Hinweise zum Autograding

Die Übungsaufgaben werden in ***hashtable.cpp*** gelöst und in *main.cpp* getestet. Alle anderen Dateien bleiben unberührt! Zeile 2 in *main.cpp* - #define TEST 1 - bestimmt ob Tests ausgeführt werden, oder nicht. 0 bedeutet, dass nicht getestet wird. Für die Abgabe muss die Konstante immer den Wert 1 aufweisen.

In der *main* Funktion dürfen **keine eigenen return statements** gesetzt werden (K.O. Kriterium). Sonst werden keine Testfälle ausgeführt.

Entfernen Sie vor der automatischen Bewertung **sämtliche Eingabeaufforderungen (scanf, gets,...)** von der Konsole. Diese führen zu einem Timeout und verhindern das Autograding auf Github Classroom.

Übungsaufgabe

Implementierung Hashtabelle
(10 Classroom Punkte)

Implementieren Sie eine einfache Hashtabelle mit Hilfe von Listen (Implementierung steht im Repository bereits zur Verfügung). Gegeben ist eine Headerdatei hashtable.hpp, welche folgende Methoden deklariert:

```
void ht_put(hashtable * ht, char * key, char * value);
```

Legt einen gegebenen Wert (entspricht *station_name* aus der Struktur *element* der Listenimplementierung) unter dem zugehörigem Schlüssel (entspricht *icao_code* in der Struktur element) in der Hashtabelle ab. Ist ein Schlüssel in der Hashtabelle vorhanden, so soll der Wert ersetzt werden. Wird für *value* NULL übergeben, so soll der Schlüssel gelöscht werden. (Anm.: Vom Schlüssel und Wert muss keine Kopie angelegt werden. Für das richtige Speichermanagement ist in diesem Fall die Liste verantwortlich.)

```
char * ht_get(hashtable * ht, char * key);
```

Liefert einen Zeiger auf den Wert (*station_name* aus der Struktur *element* der Listenimplementierung) zum übergebenen Schlüssel zurück.

```
int ht_hash(hashtable * ht, char * key);
```

Liefert für einen beliebigen Schlüssel eine Ganzzahl im entsprechenden Wertebereich zurück. Überlegen

Sie sich hierfür eine geeignete Hashfunktion.

In der Header Datei ist weiters die folgende Struktur definiert:

```
struct _hashtable {
     int capacity;   Größe der Hashtabelle
     list ** buckets;   Zeiger auf Array von Listenzeigern
};
```

Das Hauptprogramm liest ca. 5500 Datensätze von Wetterstationen (Code + Name) aus der Datei „stations.csv" ein und trägt diese in die Hash-Tabelle ein.

Hinweis: Zur Überprüfung Ihrer Hashbunkton können Sie die Größe der einzelnen Buckets auf der Konsole ausgeben. So sehen Sie, ob die gewählte Hashfunktion für dieses konkrete Beispiel gut geeignet ist. Je besser Ihre Hashfunktion die Schlüssel auf die vorhandenen Buckets aufteilt, desto mehr Punkte können Sie erreichen.