FH CAMPUS WIEN
UNIVERSITY OF APPLIED SCIENCES

Algorithmen & Datenstrukturen ILV
Sortieralgorithmen

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:

- Sortieralgorithmen
- Rekursion

Hinweise zum Autograding

Die Übungsaufgaben werden in ***sort.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 1

Selection Sort
(5 Punkte)

Implementieren Sie den Selection Sort Algorithmus. Es soll alphabetisch aufsteigend nach *station_name* sortiert werden.

Übungsaufgabe 2

Merge Sort
(5 Punkte)

Implementieren Sie den Merge Sort Algorithmus. Es soll alphabetisch absteigend nach *station_name* sortiert werden.

Übungsaufgabe 3

Function Pointer und Selection Sort
(5 Punkte)

Erweitern Sie Selection Sort dahingehend, dass der Algorithmus universell hinsichtlich der zu vergleichenden Eigenschaft ist. In unserem Beispiel ist das die Sortierung nach *station_name* oder *icao_code*. Dabei

soll ein Function Pointer zum Einsatz kommen. Die Funktion Selection Sort könnte dann folgenden Prototypen aufweisen:

void selection_sort_fp(**element*** stations, **int** size, **int** (*comp)(**element** *, **element***));

stations – das übergebene, zu sortierende Array von Flugplätzen
size – Größe des Arrays
comp – Funktionszeiger auf eine Vergleichsfunktion

Zur Veranschaulichung der Funktionsweise Implementieren Sie eine Funktion

int compareByIcaoCode(**element*** a, **element** * b);

, welche die übergebenen Elemente nach dem *icao_code* lexikalisch vergleicht. Die Funktion soll 0 zurückliefern, wenn die *icao_codes* der Elemente a und b ident sind, > 0, wenn der i*cao_code* von a lexikalisch nach dem *icao_code* von b gereiht werden soll und < 0, wenn der *icao_code* von a lexikalisch vor dem *icao_code* von b gereiht werden soll.