Freiwillige Offline-Aufgabe 09-02 (INF & WI): Templates: Funktion `sort_three_vars()` *(geübte C++ Konstrukte: Funktions-Templates & deren explizite Instanziierung)*

Schreiben Sie eine Template-Funktion `void sort_three_vars(…)`, welche die Werte ihrer drei Parametervariablen aufsteigend sortiert.

Die Funktion `sort_three_vars()` soll daher drei Parameter `x1`, `x2` und `x3` haben vom identischen, allgemein gehaltenen Typ `T`.

Schreiben Sie für den Prototypen der Template-Funktion eine geeignete, vollständige Headerdatei `sort_three_vars.h`. Schreiben Sie ferner eine Implementierungsdatei `sort_three_vars.cpp`, welche die vollständige Realisierung der Funktion sowie die *explizite Instanziierung* für `int` und für `string` realisiert.

Programmieren Sie darüber hinaus in einer Datei `main.cpp` ein Hauptprogramm, welches die angegebenen Testläufe realisiert.
***Kommentar**: In diese Datei brauchen (wie auch in der Vorlesung dargestellt) keine `extern` Anweisungen zur Verhinderung einer zusätzlichen impliziten Instanziierung eingefügt zu werden: Da der Code in `main.cpp` nur die Headerdatei inkludiert und dort nur der **Prototyp** der Template-Funktion steht, "bleibt der Rumpf dieser Funktion unbekannt" und der Code in `main.cpp` kann gar keine implizite Instanziierung durchführen. `extern` Anweisungen wären also unnötig und führen bei der Art, wie wir aktuell die Jenkins-Tests für diese Aufgabe programmiert haben, sogar zu Fehlermeldungen …
Ein Vorteil der **expliziten** Instanziierung (auf die beschriebene Art) ist ja insbesondere, dass man sich diese `extern` Anweisungen sparen kann …*

***Achtung!** Im Hauptprogramm werden Sie zuerst drei `int` Eingaben mittels `cin >> …` entgegennehmen, dann drei `string` Eingaben mittels `getline()`. Programmieren Sie daher auf jeden Fall hinter die zweite `cin >> …` Eingabe (und wenn Sie es systematisch machen wollen hinter jede `cin >> …` Eingabe) den Befehl `cin.ignore()`, sonst funktioniert die anschließende (erste) `getline()` Eingabe nicht wie gedacht, da noch ein `Newline` (Zeilenumbruch) auf der Eingabe verbleibt.*
Siehe GIP-INF Übung am 8.11.2019!

Testläufe (Benutzereingaben zur Verdeutlichung unterstrichen):

```
Bitte geben Sie die erste int Zahl ein: ? 2
Bitte geben Sie die zweite int Zahl ein: ? 3
Bitte geben Sie die dritte int Zahl ein: ? 1
Erste Zahl nachher: 1
Zweite Zahl nachher: 2
Dritte Zahl nachher: 3
Bitte geben Sie den ersten String ein: ? b
Bitte geben Sie den zweiten String ein: ? c
Bitte geben Sie den dritten String ein: ? a
Erster String nachher: a
Zweiter String nachher: b
Dritter String nachher: c
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie die erste int Zahl ein: ? 6
Bitte geben Sie die zweite int Zahl ein: ? 5
Bitte geben Sie die dritte int Zahl ein: ? 4
Erste Zahl nachher: 4
Zweite Zahl nachher: 5
Dritte Zahl nachher: 6
Bitte geben Sie den ersten String ein: ? zz zz
Bitte geben Sie den zweiten String ein: ? yy yy
Bitte geben Sie den dritten String ein: ? xx xx
Erster String nachher: xx xx
Zweiter String nachher: yy yy
Dritter String nachher: zz zz
Drücken Sie eine beliebige Taste . . .
```

```
Bitte geben Sie die erste int Zahl ein: ? 1
Bitte geben Sie die zweite int Zahl ein: ? 2
Bitte geben Sie die dritte int Zahl ein: ? 3
Erste Zahl nachher: 1
Zweite Zahl nachher: 2
Dritte Zahl nachher: 3
Bitte geben Sie den ersten String ein: ? z
Bitte geben Sie den zweiten String ein: ? y
Bitte geben Sie den dritten String ein: ? (leerer Eingabestring)
Erster String nachher: (leere Ausgabe, aber ein Leerzeichen hinter dem Doppelpunkt!)
Zweiter String nachher: y
Dritter String nachher: z
Drücken Sie eine beliebige Taste . . .
```
