FH-OÖ Hagenberg/HSD
Software Entwurf 2, SS 2013
Übung 8

Name: Abgabetermin: KW 22

Mat.Nr: Punkte:

Übungsgruppe: korrigiert:

Aufwand in h:

Beispiel 1 (6 Punkte) Template-Funktion: Implementieren Sie eine Template-Funktion

```
template <typename TItor, typename TValue>
Shift(TItor begin, TItor end, size_t count, TValue val);
```

die alle Elemente im über die Iteratoren `begin` und `end` gegebenen Bereich um `count` Stellen nach vorne verschiebt. Die letzten `count` Elemente sollen mit dem Wert `val` aufgefüllt werden. Es muss nicht geprüft werden, ob `count` größer als die Anzahl der Elemente ist. Achten Sie darauf, dass Ihr Algorithmus auch für Container mit sequentiellem Zugriff funktioniert, die Effizienz für Container mit wahlfreiem Zugriff aber darunter nicht leidet. Verwenden Sie zu diesem Zweck die Funktion `std::advance` (im Header `<iterator>`).

Beispiel 2 (8 Punkte) Generische Prüfung auf Sortierung: Implementieren Sie einen generischen Algorithmus, der die Sortierung eines über Iteratoren gegebenen Bereichs in einem Container prüft. Folgende 4 Fälle sollen dabei unterschieden werden:

- Der Bereich ist aufsteigend sortiert.
- Der Bereich ist absteigend sortiert.
- Alle Elemente sind gleich oder der Bereich ist leer.
- Der Bereich ist unsortiert.

Definieren Sie einen entsprechenden Enumerationstypen und verwenden diesen als Rückgabetyp für Ihren Algorithmus.

Implementieren sie weiters eine überladene Variante mit einem Prädikat als dritten Parameter, das die für die Sortierung relevante Vergleichsfunktion angibt.

Achten Sie darauf, dass Ihr Algorithmus auch für Container mit sequentiellem Zugriff funktioniert.

Beispiel 2 (10 Punkte) Laufzeitkomplexität: Bestimmen Sie für folgende rekursiven Funktionen die entsprechende Laufzeitkomplexität. Studieren Sie dazu die zugehörigen Vorlesungsunterlagen!

1. $T(n) = 3T(\frac{n}{2}) + n^2$
2. $T(n) = 12T(\frac{n}{4}) + n$
3. $T(n) = 5T(\frac{n}{5}) + \frac{n}{10}$
4. $T(n) = 8T(\frac{n}{4}) + \log n$
5. $T(n) = 2T(\frac{n}{2}) + n^2$
6. $T(n) = 27T(\frac{n}{3}) + 2n^3$

Allgemeine Hinweise: Legen Sie bei der Erstellung Ihrer Übung großen Wert auf eine **saubere Strukturierung** und auf eine **sorgfältige Ausarbeitung!** Verwenden Sie immer **Module**, um den Testtreiber und die eigentliche Implementierung zu trennen! Dokumentieren Sie alle Schnittstellen und versehen Sie Ihre Algorithmen an entscheidenden Stellen ausführlich mit **Kommentaren! Testen** Sie ihre Implementierungen ausführlich! Geben Sie **Lösungsideen** an!