Thema, Ziele: Listen, Algorithmen und Komplexitätstheorie

Aufgabe 1: Klasse für die Speicherung von Messwerten

Als Vorgabe erhalten Sie eine Templateklasse für eine einfach verkettete Liste. Implementieren Sie darauf basierend eine Klasse für die Verwaltung von Messreihen. Die Klasse speichert eine beliebige Anzahl von Messwerten. Einer zu implementierenden Methode können Sie einen Toleranzwert in Prozent übergeben. Die Methode entfernt dann aus der Liste alle Messwerte, die mehr als dieser Toleranzwert vom Mittelwert der Messwerte abweichen.

Überlegen Sie sich als erstes, ob die Beziehung zur Listenklasse eine Vererbung oder eine Aggregation ist. Begründen Sie Ihre Wahl.

Aufgabe 2: Komplexitätsbetrachtungen

Bestimmen Sie den benötigten Aufwand der folgenden Algorithmen in O-Notation. Alle Algorithmen berechnen die Potenz $c=a^b$

Hinweis:

Die Aufgabe ist einfacher zu lösen, wenn Sie die Algorithmen mit einfachen Zahlenwerten nachvollziehen. Die Algorithmen selbst müssen Sie nicht verstehen.

```
// berechnet a^b, Voraussetzung: b>=0
double potenz1(double a, int b)
{
  double c = 1;
  while (b > 0)
  {
    c = c * a;
    b = b - 1;
  }
  return c;
}

// berechnet a^b, Voraussetzung: b>=0
double potenz2(double a, int b)
{
  if (b == 0)
    return 1;
  else
    return potenz2(a, b-1) * a;
}

// berechnet a^b, Voraussetzung: b>=0
double potenz3(double a, int b)
{
  double c = 1;
  while (b > 0)
  {
    if (b % 2 == 1)
      c = c * a;

    a = a * a;
    b = b / 2;
  }
  return c;
}
```

Aufgabe 3: Implementation eines dynamischen Stacks mit Hilfe einer verketteten Liste (optional)

Ein Stack (Stapel, LIFO, Last-In-First-Out) ist eine Datenstruktur, bei der das Element, das zuerst auf den Stack gespeichert wird, als letztes wieder ausgelesen wird.

Implementieren Sie eine Klasse Stack, die nicht auf Arrays, sondern mit der vorgegebenen einfach verketteten Liste arbeitet. Die Stack-Klasse soll als Template implementiert sein.

Überlegen Sie sich auch hier als erstes, ob die Beziehung zur Listenklasse eine Vererbung oder eine Aggregation ist. Begründen Sie Ihre Wahl.

Als Methoden müssen Sie die bekannten Stackoperationen `push()`, `pop()`, `isEmpty()` und `peek()` realisieren. Reservieren Sie im Konstruktor die als Parameter übergebene Anzahl Listenelemente. Falls ein `push()` bei einem vollen Stack probiert wird, soll jetzt aber nicht ein Fehler ausgegeben werden, stattdessen sollen Sie den Stack dynamisch um zusätzliche Elemente vergrössern.