Human-Computer Interaction Group

Leibniz Universität Hannover

Programmieren 1 - WS 2015/16

Prof. Dr. Michael Rohs, Henning Pohl, M.Sc., Oliver Beren Kaul, M.Sc.

Übungsblatt 13

Dieses Übungsblatt soll bei der Vorbereitung auf den Abschlusstest helfen. Ihre Lösung soll nicht dem Tutor vorgestellt und nicht in das Abgabesystem hochgeladen werden. Schauen Sie sich zur Vorbereitung auf den Abschlusstest auf jeden Fall auch erneut die vorhergehenden Übungen sowie die Vorlesungsfolien an.

Allgemeine Hinweise zur Kompilierung

Die Programming I Java Library ist als jar-Datei in assignment13.zip enthalten. Die Kompilierung erfolgt auf der Kommandozeile. Die Programming I C Library muss, wie bei den vorhergehenden C-Übungen, über den Pfad erreichbar sein. Im Abschlusstest werden alle notwendigen Bestandteile in kompilierter Form vorhanden sein. Die Dokumentation wird ebenfalls zur Verfügung stehen.

Die Dokumentation der Programming I C Library finden sie unter:

http://hci.uni-hannover.de/files/prog1lib/files.html

Die Dokumentation der Programming I Java Library finden sie unter:

http://hci.uni-hannover.de/files/prog1javalib/index.html

Aufgabe 1: four_sorted_digits.c

Die Template-Datei für diese Aufgabe ist `four_sorted_digits.c`. Implementieren Sie die Funktion `bool four_sorted_digits(String s)`. Diese Funktion soll `true` zurückgeben, wenn `s` mindestens 4 hintereinander stehende und aufsteigend sortierte Dezimalziffern enthält. Sonst gibt die Funktion `false` zurück.

Hinweis: Zur Erinnerung: String ist definiert als typedef String char*. Ein String ist ein char-Array mit terminierendem 0-Zeichen. Daher können Indizes verwendet werden, um auf die einzelnen Buchstaben zuzugreifen: `s[i]`.

Anweisungen zum Kompilieren und Ausführen finden Sie in der Template-Datei.

Aufgabe 2: center_or_zero.c

Die Template-Datei für diese Aufgabe ist `center_or_zero.c`. Implementieren Sie die Funktion `double center_or_zero(DoubleList *list)`. Diese Funktion soll das Element an der mittleren Position der List zurückgeben. Wenn die Liste eine gerade Anzahl an Elementen enthält, soll die Funktion den Wert 0 zurückgeben. In dieser Aufgabe dürfen die Listen der prog1lib nicht verwendet werden.

Anweisungen zum Kompilieren und Ausführen finden Sie in der Template-Datei.

Aufgabe 3: NodesEqualToParent.java

Die Template-Datei für diese Aufgabe ist `NodesEqualToParent.java`. Implementieren Sie die Methode `int numberOfNodesThatAreEqualToTheirParent()`. Diese Methode soll die Anzahl der Knoten des Binärbaums zurückgeben, die den gleichen Wert haben, wie ihr Elternknoten.

Hinweis: Es kann zweckmäßig sein, hierfür eine (rekursive) Hilfsmethode in der Klasse `Tree` oder der Klasse `Node` zu implementieren.

Anweisungen zum Kompilieren und Ausführen finden Sie in der Template-Datei.

Aufgabe 4: IsSearchTree.java

Die Template-Datei für diese Aufgabe ist `IsSearchTree.java`. Implementieren Sie die Methode `boolean isSearchTree()`. Diese Methode soll `true` zurückgeben, wenn es sich um einen Suchbaum handelt. Andernfalls soll sie `false` zurückgeben.

Hinweis: Ein binärer Baum ist dann ein Suchbaum, wenn für jeden Knoten gilt, dass die Werte im linken Unterbaum alle kleiner sind als der Wert des Knotens und die Werte im rechten Unterbaum alle größer sind als der Wert des Knotens.

Hinweis: Es kann zweckmäßig sein, hier eine (rekursive) Hilfsmethode in der Klasse `SearchTree` oder der Klasse `SearchNode` zu implementieren.

Anweisungen zum Kompilieren und Ausführen finden Sie in der Template-Datei.