Universität Regensburg

PD Dr. J. Reischer 4.2.2014

Klausur "ADP" WS 2013/2014

Nachname, Vorname	
Abschluss (BA, MA, FKN etc.)	
Matrikelnummer, Semester	
Versuch (1/2/3)	

Bitte füllen Sie zuerst den Kopf des Angabenblattes aus!

Die Klausur dauert 90 Minuten.

Es sind maximal 40 Punkte zu erreichen.

Es sind keine Hilfsmittel zugelassen.

Bitte beantworten Sie alle Fragen direkt auf das Angabenblatt.
Nutzen Sie ggf. die Rückseite und kennzeichnen Sie dies entsprechend.
Eigene Schmierblätter sind nicht erlaubt.

Bei mehreren oder mehrdeutigen Lösungen wird die schlechtere Lösung gewertet. Streichen Sie daher ungültige Lösungen eindeutig durch.

Verwenden Sie nur Java, C# oder Pseudokode für Programmieraufgaben.

Viel Erfolg!

Aufgabe 1: *Iteration Analyse* (4 Punkte)

Konstruieren Sie für folgenden Algorithmus in Pseudokode ein Struktogramm.

```
string InsertSort(string S)
{
  char C;
  int32s I, K, N := S.Length;
  for I := 1 upto N-1
  {
    C := S[I]; K := I;
    while (K > 0) & (S[K-1] > C)
    {
      S[K] := S[K-1];
      K := K-1;
    }
    do;
    S[K] := C;
  }
  do;
  return S;
};
```

Aufgabe 2: *Iteration Synthese* (10 Punkte)

2.1: Schreiben Sie eine Funktion **char**`[][] StringToArray(`**string** `S)`, die den String `S` in ein *quadratisches* Array mit den Ausmaßen der Stringlänge von `S` einträgt. Der String soll nur horizontal oben (Zeile 0), vertikal links (Spalte 0) und diagonal von links oben nach rechts unten eingetragen werden; einige Felder des Arrays bleiben daher unbesetzt. Das Array müssen Sie selbst anlegen und als Funktionsergebnis zurückgeben; eine gesonderte Initialisierung der **char**-Werte ist nicht notwendig, da diese automatisch mit dem Unicode-Wert 0 vorbelegt werden. Sie dürfen davon ausgehen, dass `S` nicht **null** ist. (8 Punkte)

2.2: Schreiben Sie zusätzlich eine Funktion **int32s** `UnsetChars(`**char**`[][] C)`, die ermittelt, wie viele Felder im Array `C` nach Ausführung von `StringToArray("…")` noch unbesetzt sind (d. h. noch den Initialisierungswert 0 aufweisen). (2 Punkte)

Hinweis: Die beiden Aufgaben können unabhängig voneinander gelöst werden!

Beispiel für S = "ABC":

'A'	'B'	'C'
'B'	'B'	
'C'		'C'

2 unbesetzte Felder

Beispiel für S = "1234":

'1'	'2'	'3'	'4'
'2'	'2'		
'3'		'3'	
'4'			'4'

6 unbesetzte Felder

(Fortsetzung Aufgabe 2)

Aufgabe 3: *Iteration Synthese* (10 Punkte)

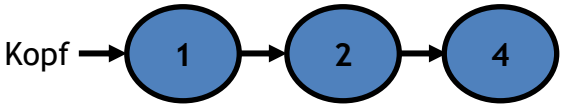
Schreiben Sie eine Funktion `ListNode ArrayToList(int32s[] A)`, die aus einem Array `A` eine *unidirektionale Liste* erzeugt. Die Liste enthält so viele Knoten wie das Array Elemente. Der jeweilige Wert von `A[…]` soll als Knoteninhalt für den entsprechenden Knoten eingesetzt werden. Der erste Knoten (Listenkopf) ist als Funktionsergebnis zurückzugeben. Sie dürfen davon ausgehen, dass `A ≠ null` ist.

Definition von `ListNode`:

```
class ListNode
{ // Datenfelder
  public int32s Cont;
  public ^ListNode SuccNode;
  // Konstruktor
  constr ListNode(string InitCont)
  {
    Cont := InitCont; SuccNode := null;
  };
};
```

Beispiel:

`A[0] = 1, A[1] = 2, A[2] = 4`



Aufgabe 4: *Rekursion Analyse* (10 Punkte)

Gegeben ist folgende rekursive Funktion F in Pseudokode:

```
// Rekursive Funktion F
int32s F(int32s N)
{
  if N ≤ 0 then
    return 1;
  else if N ≤ 2
    return 2*N + 1;
  else
    return F(N–1) + F(N–2) – F(N–3);
};
```

4.1: Welche Werte liefert die Funktion F für die Aufrufe N = 0, 1, 2, 3, 4, 5, 6 zurück? Tragen Sie die Ergebnisse für F in unten stehende Tabelle ein (2 Versuche, ansonsten eigene Tabelle; streichen Sie einen Fehlversuch deutlich durch). (7 Punkte)

N	0	1	2	3	4	5	6
F(N)							
F(N)							

4.2: Welchen Komplexitätsgrad bezüglich des Laufzeitverhaltens O(?) weist die Funktion F auf (Begründung!)? (2 Punkte)

4.3: Welche Formen der Rekursion treten hier auf? (1 Punkt)

Aufgabe 5: *Rekursion Synthese* (6 Punkte)

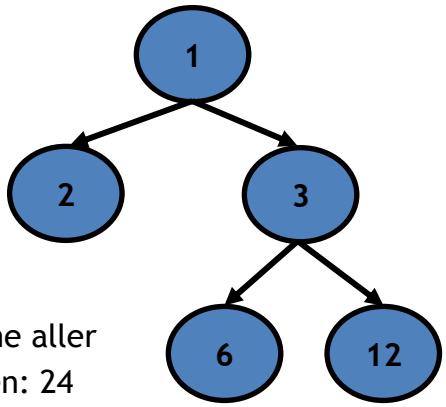
Schreiben Sie eine Methode **int** SumNodes(TreeNode Node) als *rekursive Funktion* mit ganzzahligem Rückgabewert, die für den gesamten *binären Baum* ab inklusive Node die Summe aller Knoteninhalte (Cont) ermittelt. Der Baum kann beliebig groß sein. Es darf keine Schleife vorkommen.

Rekursionslogik (deskriptiv):
Die Summe des Teilbaums ab einschließlich Node ist der Wert von Cont in Node selbst plus die Summe der Cont-Werte des linken Teilbaumes plus die Summe der Werte des rechten Teilbaumes. Achten Sie auf Sonder-/Grenz- bzw. Terminationsfälle.

Definition eines Knotens:

```
class TreeNode
{ // Datenfelder
  public int32s Cont;
  public ^TreeNode LeftNode,RghtNode;
  // Konstruktor
  constr TreeNode(int32s InitCont)
  {
    Cont := InitCont;
    LeftNode := null;
    RghtNode := null;
  }
}
```

Beispiel:



Summe aller Knoten: 24