Universität Regensburg

PD Dr. J. Reischer 26.07.2012

Klausur "ADP" SS 2012

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 45 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* (5 Punkte)

Konstruieren Sie für folgenden Algorithmus in Pseudokode ein Struktogramm. Die Methode `String.Exchange(ZeichenIndex1,ZeichenIndex2)` kann als gegeben betrachtet werden.

```
string BubbleSort(string S)
{
  boole Sort;
  int32s I, N := S.Length;
  do
  {                              // false, solange in einem Durchlauf
    Sort := false;               // nichts sortiert wurde
    for I := 1 upto N-1          // jedes Element testen, ob korrekt
      if S[I-1] > S[I] then      // einsortiert; wenn falsche Reihung:
      {                          // Element muss getauscht werden
        S.Exchange(I-1,I);       // Tausch der Zeichen an Indexen I/I-1
        Sort := true;            // anzeigen, dass noch etwas sortiert
      };                         // werden musste (weitere Sortierrunde
    do;                          // notwendig)
  }
  while Sort;                    // weiter, solange sortiert wurde
  return S;                      // umsortierter String zurück
};
```

Aufgabe 2: *Iteration Synthese/Analyse* (10 Punkte)

2.1: Schreiben Sie eine Funktion `boole IsSymmetricMatrix(char[][] Matrix)`, die eine quadratische Matrix von Zeichen daraufhin testet, ob die Matrix bezüglich der Diagonale von links oben nach rechts unten symmetrische Werte besitzt (s. Beispiele); d. h. das obere und untere Dreieck bezüglich der Diagonalen enthält gespiegelt die gleichen Werte. Die Matrix kann dabei beliebige Ausdehnung außer 0×0 besitzen: 1×1, 2×2, 3×3, …, N×N. Die Quadratigkeit der Matrix muss hier nicht getestet werden.

(8 Punkte)

symmetrisch (Ergebnis **`true`**):

A	B	C
B	C	D
C	D	E

A	B	C	D
B	C	D	E
C	D	E	F
D	E	F	G

nicht-symmetrisch (Ergebnis **`false`**)

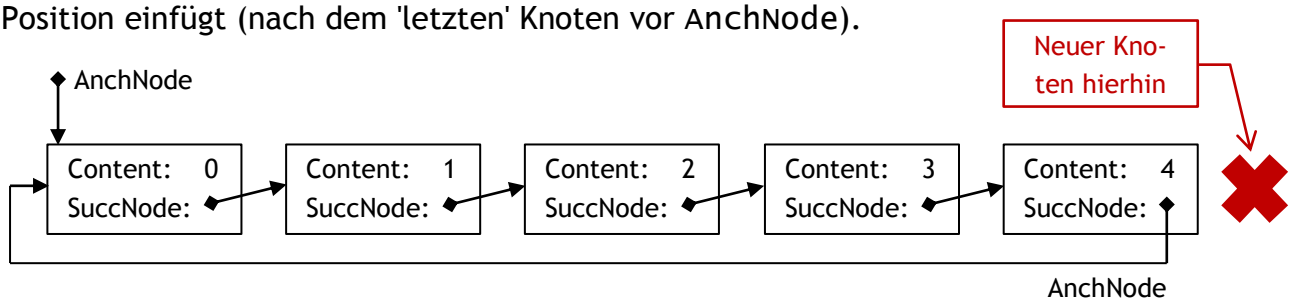
A	X	C
B	C	D
C	D	E

A	B	C	D
B	C	X	E
C	D	E	F
D	E	F	G

2.2: Welchen Komplexitätsgrad bezüglich Laufzeitverhalten O(?) weist die Funktion auf (Begründung!)? (2 Punkte)

Aufgabe 3: *Iteration Synthese* (10 Punkte)

Schreiben Sie eine Funktion `void AppendNode(ListNode AnchNode, int32s NodeCont)`, die in eine zu einem Ring geschlossene Liste beliebiger Länge (mit mind. 1 Element) einen neuen Knoten mit dem Inhalt `NodeCont` an der mit X gekennzeichneten Position einfügt (nach dem 'letzten' Knoten vor `AnchNode`).



Beachten Sie, dass `AnchNode` auf den ersten Knoten zeigt und der letzte Knoten *vor* `AnchNode` ebenfalls auf `AnchNode` verweist (dadurch wird die Liste erst zum Ring).
Es darf vorausgesetzt werden, dass `AnchNode` $\neq$ `null` ist, d. h. nur einer Ringliste mit mindestens einem Listenelement kann ein neues Element am Ende hinzugefügt werden.
Die Referenzvariable `AnchNode` zeigt nach der Operation unverändert auf Knoten 0.

Aufgabe 4: *Rekursion Analyse* (10 Punkte)

Gegeben ist folgende rekursive Funktion in Pseudokode:

```
int32s F(int32s N)
{
  if (N = 0) then
    return 0;
  else
    return F(N–1)+N;
};
```

4.1: Welche Werte liefert die Funktion für die Aufrufe N = 0, 1, 2, 3, 4, 5 zurück? Erzeugen Sie hierzu eine Tabelle mit den sechs Eingabewerten für N und den entsprechenden Rückgabewerten für F(N). (6 Punkte)

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

4.3: Nennen Sie zwei verschiedene Klassifikationsdimensionen für die in obigem Kode auftretenden Arten der Rekursion. (2 Punkte)

Aufgabe 5: *Rekursion Synthese* (10 Punkte)

Schreiben Sie eine Methode `int32s CountCapitals(string S)` als *rekursive Funktion* mit ganzzahligem Rückgabewert, die von einer beliebigen Zeichenkette `S` die darin enthaltene Anzahl *englischer* Großbuchstaben 'A' bis 'Z' zählt (ob die Funktion `static`, `public` usw. ist, soll hier nicht interessieren). Rekursionslogik:

Die Anzahl der Großbuchstaben insgesamt (auf einer Rekursionsebene!) ermittelt sich aus der Anzahl Großbuchstaben des Kopfstrings von `S` (der jeweils nur aus dem ersten Zeichen besteht), plus der Anzahl der Großbuchstaben aus dem Reststring von `S` (ab dem 2. Zeichen bis Ende):

Auf jeder Rekursionsebene wird immer nur das erste Zeichen des aktuellen `S` daraufhin getestet, ob es ein Großbuchstabe ist oder nicht; alle anderen Großbuchstaben werden jeweils auf einer anderen Rekursionsebene ermittelt. Achten Sie zudem auf Grenzfälle bzw. Terminationsbedingungen.

Sie dürfen folgende Funktionen zur Ermittlung eines Teilstrings T bzw. eines Zeichens C aus S verwenden:

C#:
```
T = S.Substring(StartIndex,AnzahlZeichen);
C = S[ZeichenIndex];
```
Java:
```
T = S.substring(BeginIndex,EndIndex_Exklusive);★
C = S.charAt(ZeichenIndex);
```
Pseudokode:
```
T := S.Extract(StartIndex,AnzahlZeichen);
C := S[ZeichenIndex];
```

(★ Exklusiver Endindex: Index des ersten Zeichens, das *nicht* mehr zu T gehören soll!)

Sie dürfen hingegen *nicht* verwenden: `Character.isUpperCase(char c)`! Bitte umschreiben Sie die gewünschte Funktionalität mit einem eigenen Ausdruck.