Universität Regensburg

PD Dr. J. Reischer 24.07.2013

Klausur "ADP" SS 2013

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.

```
boole IsPrime(int32u N)
{
  boole[N + 1] Primes;
  int32u K, X;
  for K := 2 upto N Primes[K] := true; do;
  for K := 2 upto N
    if Primes[K] then
    {
      X := K • K;
      while X ≤ N
      {
        Primes[X] := false;
        X := X + K;
      }
      do;
    };
  do;
  return Primes[N];
};
```

Aufgabe 2: *Iteration Synthese* (10 Punkte)

2.1: Schreiben Sie eine Funktion `int SearchWord(char[][] M, string S)`, die *alle* Vorkommnisse des Strings S horizontal und/oder vertikal in der *quadratischen* Matrix M ermittelt und die Anzahl zurückgibt (nicht diagonal); d. h. die Zeichen aus S müssen direkt hintereinander in der Zeichenmatrix auftreten (s. Beispiele unten). *Der Suchstring kann auch mehrfach überkreuzend und/oder überlappend in einer Zeile bzw. Spalte vorkommen*. Die Matrix kann beliebige Ausdehnung außer 0×0 besitzen; die Quadratigkeit der Matrix mit einer Größe > 0×0 darf vorausgesetzt werden. Benutzen Sie `Matrix.length` [Java] bzw. `Matrix.Length` [C#, Pseudokode] zur Ermittlung der horizontalen/vertikalen Matrixgröße; weitere API-Methoden sind erlaubt. (8 Punkte)

2.2: Schreiben Sie zusätzlich eine Funktion `boole Contains(char[][] M, string S)`, die feststellt, ob ein Suchstring S mindestens 1 Mal irgendwo in M vorkommt. (2 Punkte)

Beispiel für 3×3:

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

"1": 2 Mal enthalten (!)
"22": 1 Mal enthalten
"33": 3 Mal enthalten
"3333": 0 Mal enthalten

Beispiel für 4×4:

'A'	'N'	'N'	'A'
'N'	'O'	'O'	'N'
'N'	'O'	'O'	'N'
'A'	'N'	'N'	'A'

"A"/"N": 8 Mal enthalten (!)
"NN"/"NA": 4 Mal enthalten
"NOON": 4 Mal enthalten
"ANN": 4 Mal enthalten
"AOOA": 0 Mal enthalten

Hinweis: Der Leerstring "" ist genau so oft enthalten, wie es Felder in der Matrix gibt!

(Fortsetzung Aufgabe 2)

Aufgabe 3: *Iteration Synthese* (10 Punkte)

Schreiben Sie eine Funktion `boole AreEqual(ListNode Head1, ListNode Head2)`, die zwei Listen auf Inhaltsgleichheit prüft, d. h. alle Content-Inhalte müssen in den beiden Listen *paarweise gleich* sein (der String*wert* soll hierbei verglichen werden). `Head1` ist der Einstiegsknoten in die erste, `Head2` in die zweite Liste. Die Längen der Listen sind unbekannt und können *nicht* vorausgesetzt werden. Sie können wie in der Übung davon ausgehen, dass Getter-Methoden zum Zugriff auf die Listenknoten-Member vorhanden sind (Sie können aber auch direkt darauf zugreifen, da die Member unten als `public` markiert sind und Sie dies in Ihrer Sprache ebenfalls voraussetzen können).

Definition von `ListNode`:

```
class ListNode
{
     public string Content := "";
     public ListNode SuccNode := null;
     constr ListNode(string Cont)
     {
          Content := Cont;
     };
};
```

Aufgabe 4: *Rekursion Analyse* (10 Punkte)

Gegeben sind folgende rekursive, zusammengehörige Funktionen in Pseudokode:

```
// Rekursive Funktion 1
int32s F1(int32s N)
{
  if N = 0 then
    return F2(N);
  else
    return F2(N – 1);
};

// Rekursive Funktion 2
int32s F2(int32s N)
{
  if N = 0 then
    return N + 1;
  else
    return F1(N % 2);           // % ist Modulo-Operator
};
```

4.1: Welche Werte liefert die Funktion F1 für die Aufrufe N = 0, 1, 2, 3, 4, 5 zurück? Tragen Sie die Ergebnisse für F1 (und als optionale Zwischenrechnung für Sie evtl. auch F2) in unten stehende Tabelle ein. (6 Punkte)

N	0	1	2	3	4	5
F1(N)						
F2(N)						

4.2: Welchen Komplexitätsgrad bezüglich des Laufzeitverhaltens O(?) weist die Funktion F1 auf (Begründung!)? (2 Punkte)

4.3: Welche Formen der Rekursion treten hier auf? (2 Punkte)

Aufgabe 5: *Rekursion Synthese* (10 Punkte)

Schreiben Sie eine Methode `int Modulo(int N, int M)` als *rekursive Funktion* mit ganzzahligem Rückgabewert, die für zwei nicht-negative Ganzzahlen N und M den Restwert der ganzzahligen Division N/M durch *fortgesetzt rekursive Subtraktion* ermittelt, d. h. N % M liefert. Es darf keine Schleife vorkommen.

Rekursionslogik (deskriptiv):
Wenn N < 0 oder M < 0: Die Funktion soll −1 zurückliefern (hier nicht beachtete Fälle für negative Parameterwerte);
M soll so lange von N rekursiv abgezogen werden, bis die Differenz negativ würde;
in diesem Fall steht der Rest fest. Division bedeutet letztlich: Wie oft passt M in N?
Wenn die Subtraktion bei 0 endet, geht die Ganzzahl-Division restfrei auf, sonst nicht.

Hinweis: Es gibt einen weiteren Sonderfall bei den Parametern N oder M, der abgefangen werden muss; als Ergebnis soll auch hier −1 zurückgegeben werden.

Beispiele:

9 % 3:	9−3−3−3 ergibt Rest 0	3 % 3:	3−3 ergibt Rest 0
8 % 3:	8−3−3 ergibt Rest 2	2 % 3:	Rest 2 (keine Subtraktion)
7 % 3:	7−3−3 ergibt Rest 1	1 % 3:	Rest 1 (keine Subtraktion)
…		0 % 3:	Rest 0 (keine Subtraktion)