Aufgabe 5 (5 Punkte): Algorithmen

Was leistet der folgende Algorithmus?

1. Sei M die Menge der ganzen Zahlen mit M = {2, 3, ..., 1000}.
2. Sei P die leere Menge, d.h. P = {}.
3. Nehme das kleinste Element m aus M und füge es in P ein.
4. Entferne m und alle ganzzahligen Vielfachen von m aus M.
5. Falls M<>{} gehe zu 3.
6. Gib die Menge P aus
7. Stop

c) [2 Punkte] Wie oft wird jeder Knoten des Baumes besucht beim Berechnen der Höhe und der Anzahl der Blätter in Ihren Moduln?

b) [9 Punkte] Entwickeln Sie ein C- Modul zur Berechnung der Anzahl der Blätter eines Binärbaums.

```
int leafes(pnode tree) { ... }
```

Aufgabe 4 (9+9+2 = 20 Punkte): Binärbäume

Gegeben sei ein Binärbaum, dargestellt durch den C-Datentyp `node` bzw. `pnode`. Die Blätter des Baums entsprechen `NULL`-Pointern bei den Söhnen `left` und `right`.

```
typedef struct node {
  int content;
  struct node* left;
  struct node* right;
} node;

typedef node* pnode;
```

Für alle knoten `n` vom Typ `node` gilt:

```
n.left->content < n.content <= n.right->content
```

a) [9 Punkte] Die Höhe eines Binärbaumes ist der maximale Abstand eines Blattes von der Wurzel. Schreiben Sie ein C-Modul zu Berechnung der Höhe eines Binärbaums.

```
int height(pnode tree) { ... }
```

d) [18 Punkte] Entwickeln Sie einen Datentyp, mit dem es möglich ist, Dreiecksmatrizen mit variabler Größe *n* zu speichern. Implementieren Sie dazu ein neues Modul `m_create(...)`, das die entsprechende Struktur aufbaut. Die Verwendung von verketteten Listen ist NICHT erforderlich!
Modifizieren Sie die Module `m_get`, `m_set`, `m_mult` und `m_add` aus Teil a) und b) entsprechend diesem neuen Datentyp. Achten Sie beim Entwurf der Datenstruktur darauf, daß die damit verbundenen Änderungen in den Modulen `m_get`, `m_set`, `m_mult`, `m_add` minimal werden. Wie werden Ihre Module in main() aufgerufen? Geben Sie dazu ein Beispiel für Variablendeklaration, Erzeugung der Matrizen und Addition! Eine Belegung der Matrizen ist für das Beispiel nicht erforderlich!

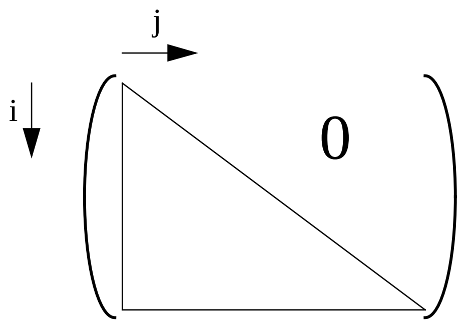
```
/* Definiert einen eigenen Datentyp dreiecksmatrix für beliebiges n */
typedef ...

/* Erzeugt eine Dreiecksmatrix der Größe n */
void m_create(...);
```

c) [2 Punkte] Welche Zeitkomplexität besitzen jeweils Ihre Module zur Addition und Multiplikation von Dreiecksmatrizen?

b) [10 Punkte] Die Menge der unteren Dreiecksmatrizen ist gegen Addition und Multiplikation abgeschlossen, d.h. die Multiplikation bzw. Addition zweier Dreiecksmatrizen ergibt wieder eine Dreiecksmatrix. Implementieren Sie Module `m_mult` und `m_add` zur Multiplikation und Addition von Dreiecksmatrizen unter Verwendung der Module `m_set` und `m_get`.

Aufgabe 3 (10+10+2+18 = 40 Punkte): C

Eine untere Dreiecksmatrix der Größe $n \times n$ ist eine Matrix, deren Elemente in der rechten oberen Hälfte Null sind, d.h. es gilt: $A_{ij} = 0$ für $i<j$:



Eine solche Matrix enthält höchstens $(n^2 + n)/2$ Elemente ungleich Null. Werden solche Matrizen in einem Array vom Typ `float matrix[n][n]` abgespeichert, so werden $(n^2-n)/2$ Speicherplätze verschwendet.

a) [10 Punkte] Entwickeln Sie eine Methode, solche Matrizen effizienter abzuspeichern, und implementieren Sie folgende Typen und Module für `const int n = 5`. Verwenden Sie zur Implementierung ebenfalls ein Array!

```
/* Definiert einen eigenen Datentyp dreiecksmatrix */
typedef ...

/* Setzt das Matrixelement (i,j) auf z */
void m_set(dreiecksmatrix d, int i, int j, float z) {...};

/* Liefert das Matrixelement (i,j) */
float m_get(dreiecksmatrix d, int i, int j) {...};
```

b) [12 Punkte] Geben Sie für die folgende Funktion und für alle Anweisungen der Funktion Vorbedingungen und Nachbedingungen und die Schleifeninvariante an. Was berechnet der Algorithmus für eine Eingabe n>=0?

```
int algo2(int n) {
int i, s;
if (n>=0) {
  i=1;
  s=1;
  do {
    s = s * i;
    i = i + 1;
  } while (i<=n);
  return(s);
}
else return(-1);
}
```

Aufgabe 2 (8+12 = 20 Punkte): Korrektheit von Algorithmen

a) [8 Punkte] Beweisen Sie, daß die folgende Funktion für beliebige ganze Zahlen p, q terminiert. Was leistet die Funktion?

```
int algo1 (int p, int q) {
int d;
if ((p>0) && q>0)) {
  d = 0;
  while (p>=0) {
    p = p - q;
    d = d + 1;
  }
  return(d);
}
else return(-1)
}
```

Aufgabe 1 (7+8 = 15 Punkte): Logik

Vereinfachen Sie die folgenden aussagenlogischen Formeln.

a)[7 Punkte] $\neg(\neg(C \wedge A) \wedge \neg(\neg C \wedge B)) \vee \neg(\neg C \vee A)$

b)[8 Punkte] $\neg F \wedge \neg G \wedge \neg(\neg A \wedge \neg(F \vee G))$

Universität Mannheim
Fakultät für Mathematik und Informatik
Lehrstuhl für Praktische Informatik IV
Prof. Dr. W. Effelsberg

Vordiplomklausur Informatik

Oktober 1996: Praktische Informatik I

Name: .... Vorname: ....

Matrikel-Nr.: .... Semester: .... Fach: ....

Hinweise:

(a) Bitte füllen Sie sofort den Kopf des Deckblatts aus.
(b) Überprüfen Sie Ihr Klausurexemplar auf Vollständigkeit (12 Seiten).
(c) Tragen Sie Ihre Lösungen soweit möglich direkt in die Klausur ein.
(d) Es sind keine Hilfsmittel zugelassen
(e) Zeit: 100 Minuten

Aufgabe	max. Punktezahl	Punkte
1	15	
2	20	
3	40	
4	20	
5	5	
Gesamt	100	