Prüfung Informatik D-MATH/D-PHYS
10. 3. 2006 9:00-11:00

Dr. Bernd Gärtner

Kandidat/in:

Name: .....

Vorname: .....

Stud.-Nr.: .....

Ich bezeuge mit meiner Unterschrift, dass ich die Prüfung unter regulären Bedingungen ablegen konnte und dass ich die untenstehenden allgemeinen Bemerkungen gelesen und verstanden habe.
Unterschrift: .....

Allgemeine Bemerkungen und Hinweise:

1. Überprüfen Sie die Vollständigkeit der ausgeteilten Prüfungsunterlagen (2 beidseitig bedruckte Blätter, bestehend aus 1 Deckseite und 3 Aufgabenseiten mit insgesamt 6 Aufgaben)!
2. Falls Sie während der Prüfung durch irgendeine Behinderung oder Störung beeinträchtigt werden, melden Sie dies bitte sofort der Aufsichtsperson! Spätere Klagen können nicht akzeptiert werden.
3. Erlaubte Hilfsmittel: **keine**.
4. Betrugsversuche führen zu sofortigem Ausschluss und können rechtliche Folgen haben.
5. Pro Aufgabe ist höchstens eine gültige Version eines Lösungsversuchs zulässig. Streichen Sie ungültige Lösungsversuche klar durch!
6. Sie dürfen die 6 Aufgaben in beliebiger Reihenfolge lösen. Konzentrieren Sie sich jeweils auf eine Aufgabe, aber teilen Sie sich Ihre Zeit ein!
7. Nach Ablauf der Prüfungszeit verlassen Sie bitte den Raum und lassen Sie nur die Blätter auf Ihrem Platz liegen, die zur Abgabe bestimmt sind! **Diese müssen alle mit Ihrem Namen beschriftet sein. Die 2 Prüfungsblätter sind dabei mit abzugeben!**
8. Die Prüfung ist bestanden, wenn Sie 56 von 111 erreichbaren Punkten erzielen.

1	2	3	4	5	6	$\sum$

Aufgabe 1. (15 Punkte) Geben Sie für jeden der folgenden Ausdrücke die logische Klammerung an! Werten Sie den jeweiligen Ausdruck in Einzelschritten aus! Das heisst, geben Sie alle Zwischenschritte der Auswertung an!

(a) `2e1+3/6/2-7`

(b) `int(3.25f*6)!=38./2||-8-1.5>3&&1.1==110/100.0f`

(c) `2-3*0.75<-(1/4)||3.159/143>0.22&&7-8>-0.9`

Aufgabe 2. (25 Punkte) Die *iterierte Quersumme* $\tilde{q}(n)$ einer natürlichen Zahl n erhält man, indem man zunächst die "normale" Quersumme $q(n)$ von n bildet (Summe aller Dezimalziffern). Falls $q(n)$ genau eine Dezimalziffer hat, so gilt $\tilde{q}(n) = q(n)$ und wir sind fertig; andernfalls ist $\tilde{q}(n)$ die iterierte Quersumme von $q(n)$, also $\tilde{q}(n) = \tilde{q}(q(n))$.
Beispiel: Für $n = 1792$ ist $q(n) = 1 + 7 + 9 + 2 = 19$, $q(19) = 1 + 9 = 10$ und $q(10) = 1 + 0 = 1$. Somit ist $\tilde{q}(n) = 1$.
Implementieren Sie eine Funktion

```
// POST: gibt die iterierte Quersumme von n zurueck.
unsigned int iterated_cross_sum(unsigned int n);
```

die für eine gegebene natürliche Zahl n den Wert $\tilde{q}(n)$ berechnet!

Aufgabe 3. (16 Punkte) Betrachten Sie die beiden Funktionen

```
bool f(unsigned int n, bool b1, bool b2)
{
  if (n == 0) return b1 && b2;
  return f(n-1, b2, !b1);
}

bool g(unsigned int n)
{
  return f(n, true, true);
}
```

und geben Sie die Nachbedingung der Funktion `g` an! Die Nachbedingung muss den Rückgabewert der Funktion in Abhängigkeit von ihren Parametern vollständig charakterisieren.

Aufgabe 4. (16 Punkte) Bezeichne im folgenden mit `a` und `b` Variablen vom Typ `int`. Im Rahmen dieser Aufgabe nehmen wir an, dass der Typ `int` ganze Zahlen exakt darstellen kann, das heisst, arithmetische Über– oder Unterläufe treten nicht auf.
Formen Sie jedes der folgenden Prädikate in ein äquivalentes Prädikat um, in welchem höchstens ein Operator (arithmetisch, logisch oder relational) auftritt! Erläutern und begründen Sie die einzelnen Schritte ihrer Umformung!

(a) `a != 4 - a && !(a < 3 * a - 7 || a >= a * a * a)`

(b) `b < 3 || a - 1 / (b - 2) < 5 / a + 1 && a >= b`

Aufgabe 5. (15 Punkte) Finden Sie die Fehler im folgenden Programm und geben Sie eine korrigierte Version des Programms an! Was macht Ihr korrigiertes Programm? Unterscheiden Sie bei der Fehlerangabe zwischen syntaktischen Fehlern (das Programm kompiliert nicht) und semantischen Fehlern (das Programm macht nicht das, was es machen soll, im Vergleich zu Ihrem korrigierten Programm).

```
 1: #include <iostream>
 2: int main
 3: {
 4:   int max = -1;
 5:   for (unsigned int i = 0; i < 5;) {
 6:     cout << "Next natural number =? ";
 7:     cin >> n;
 8:     update_max(max, n);
 9:   }
10:   cout << "Maximum is " << max << "\n";
11:   return 0;
12: }
13:
14: void update_max (int max, int n)
15: {
16:   if (n > max) max = n;
17: }
```

Aufgabe 6. (24 Punkte) Wir wollen eine Variante des Typs `unsigned int` implementieren, die einen Überlauf bei der Addition feststellt. Dazu definieren wir eine Klasse `Unsigned` mit den folgenden öffentlichen Member-Funktionen.

```
class Unsigned {
public:
  // POST: *this wurde mit x initialisiert.
  Unsigned (unsigned int x);
```

```
  // POST: y wurde zu *this addiert.
  Unsigned& operator+=(const Unsigned& y);

  // POST: Rueckgabewert ist unsigned-int-Wert von *this.
  unsigned int value() const;

  // POST: Rueckgabewert ist true genau dann, wenn *this
  //       nicht uebergelaufen ist, d.h. wenn der berechnete
  //       Wert dem mathematisch korrekten Wert entspricht.
  bool is_exact() const;

private:
  ....
};
```

Bemerkung: Ein `Unsigned` Objekt läuft insbesondere dann über, wenn eine übergelaufene Zahl zu ihm addiert wird.

(a) Ergänzen Sie den `private`-Teil, indem Sie eine geeignete Repräsentation (im Hinblick auf Teil (b)) wählen. (5 Punkte)

(b) Implementieren Sie (unter Verwendung der Repräsentation aus (a)) die vier Member-Funtionen

- `Unsigned::Unsigned (unsigned int x);` (3 Punkte)
- `Unsigned& Unsigned::operator+=(const Unsigned& y);` (10 Punkte)
- `unsigned int Unsigned::value() const;` (3 Punkte)
- `bool Unsigned::is_exact() const;` (3 Punkte)