Prüfung Informatik D-MATH/D-PHYS 26.1.2012

Dr. Bernd Gärtner, Prof. Juraj Hromkovič

Lösung.

Aufgabe 1.

Die Variable `x` ist vom Typ `int` und hat zu Beginn jeder Auswertung den Wert `8`. Die Variable `b` ist vom Typ `bool` und hat den Wert `true`.

Aufgabe	Ausdruck	Typ	Wert
a)	`6 / 3 / 4 + 1.5`	`double`	`1.5`
b)	`x != 0 && 7 / x == 0 && x / 7.0 > 1`	`bool`	`true`
c)	`7.2f - 4 > 4 && !b || b`	`bool`	`true`
d)	`b == false && ++x >= 9`	`bool`	`false`
e)	`2.0f + x++ - 2.5 * 3`	`double`	`2.5`
f)	`x % 2 % 2 - x`	`int`	`-8`
g)	`x-- % 4 == 2 && 128 % 2 == 0`	`bool`	`false`

Punktvergabe. Für jeden korrekt genannten Typ gibt es **+1 Punkt** und für jeden korrekt genannten Wert gibt es **+2 Punkte**.

Aufgabe 2.

a) 1 1 1 1 1 1 1 1 1 1 (10 Einsen)

b) 13 40 20 10 5 16 8 4 2

c) 1 4 9 16 25 36 49 64 81 100

Punktvergabe. Jeweils vier Punkte für jede Folge. -**2 Punkte** für einen Fehler (z.B. 9 oder 11 statt 10 Elemente bei a), oder ein falsches Folgenglied (Folgefehler gratis)). Bei 2 oder mehr Fehlern gibt es also keine Punkte mehr.

Aufgabe 3.
Hier ist die Standard-Lösung.

```
// POST: gibt genau dann true zurueck, wenn in der Binaerdarstellung
//       von n das Bit i gesetzt ist.
bool has_bit (unsigned int n, unsigned int i)
{
  // entferne die Bits 0,1,...,i-1
  for (int j=0; j<i; ++j) n/=2;
  // gib Bit i zurueck
  return n%2 == 1;
}
```

Alternativ könnte jemand mit den bitweisen Operatoren arbeiten, die in den Details des Skripts genannt sind.

```
bool has_bit (unsigned int n, unsigned int i)
{
  // bitwise operator solution
  return (n & (1 << i)) != 0;
}
```

Punktvergabe. **+10 Punkte** für den Versuch, in sinnvoller Weise an das i-te Bit heranzukommen, auch wenn der fehlerhaft ist. Sinnvoll heisst, dass man wohl /2 und **%2** brauchen wird. Mögliche Fehler: falsche Anzahl an Schleifendurchläufen, /2 vergessen. **+10 Punkte** für eine korrekte Lösung.

Aufgabe 4.

a) Hier sind die Fehler und ihre Korrektur.

```
int main()
{
  unsigned int dim = 10;        // Fehler 1:    const hinzufuegen
  double M[dim][dim];

  // initialisiere M mit der Einheitssmatrix
  for (int i=1; i<=dim; ++i)    // Fehler 2, 3: i=0, i<dim
    for (int j=1; j<=dim; ++j)  // Fehler 2, 3: j=0, j<dim (Wdh-Fehler)
      if (i=j)                  // Fehler 4:    i==j
        M[i][j]=1.0;
```

```
        else
          M[i][j]=0.0;

      // mache irgendetwas mit M

      return 0;
    }
```

b)
```
// transponiere M
for (int i=0; i<dim; ++i)
  for (int j=0; j<i; ++j) {
    double h = M[i][j];
    M[i][j] = M[j][i];
    M[j][i] = h;
  }
```

Punktvergabe. **+1 Punkt** für jeden gefundenen Fehler in a), und **+1 Punkt** für jede richtige Korrektur. In b), **+5** für eine doppelt geschachtelte Schleife, die alle Einträge der Matrix erreicht. **+5** für korrekte Vertauschungslogik (mit Hilfsvariable oder swap-Funktion). **+5** für korrekte Schleifenlogik. Werden nämlich wirklich alle Einträge durchlaufen, vertauscht man jeden Eintrag zweimal, was wieder zur ursprünglichen Matrix führt. Man muss also aufpassen, nur eine Hälfte der Matrix zu bearbeiten.

Aufgabe 5.
Jeder Aufruf von **hanoi** für $n > 0$ Scheiben verursacht zwei Aufrufe von **hanoi** für $n-1$ Scheiben und einen Zug. Also gilt

$$\begin{aligned} T(0) &= 0, \\ T(n) &= 2T(n-1)+1, \quad n > 0. \end{aligned}$$

Behauptung: $T(n) = 2^n - 1$.

Beweis: Das gilt für $n = 0$ (Induktionsanfang). Sei nun $n > 0$, und nehmen wir an, die Behauptung gelte für $n-1$ (Induktionsannahme). Dann können wir (Induktionsschritt) wie folgt schliessen.

$$T(n) = 2T(n-1)+1 = 2(2^{n-1}-1)+1 = 2^n - 2 + 1 = 2^n - 1.$$

Punktvergabe. **+10 Punkte** für eine korrekte Rekursionsgleichung. Maximal **+5 Punkte**, falls es Fehler gibt (+1 vergessen etc.).

+5 Punkte für eine korrekt geratene Lösung (oder eine falsche Lösung, die dafür zur Rekursionsgleichung passt).

+10 Punkte für einen formal korrekten Induktionsbeweis. -**3 Punkte**, wenn der Fall $n = 0$ nicht separat behandelt wird. -**5 Punkte** bei Rechenfehlern (Folgefehler einer falschen Rekursionsgleichung zählen nicht).

Wird rein intuitiv argumentiert ("man sieht sofort $T(0) = 0, T(1) = 1, T(2) = 3, T(3) = 7$, und dann ist allgemein $T(n) = 2^n - 1$"), so gibt es für den Teil mit dem Induktionsbeweis nur **+2 Punkte**.

Aufgabe 6.

a) Die Klasse erlaubt die wiederholte Eingabe von Fliesskommazahlen und zu jedem Zeitpunkt die Ausgabe des Mittelwerts der bisher eingegebenen Zahlen (falls überhaupt Zahlen eingegeben wurden).

b) **3.5**. Das ist der Mittelwert der beiden eingegebenen Zahlen 3 und 4.

Punktvergabe. Bei a) gibt es jeweils **+5 Punkte** für die korrekte Beschreibung der Mitgliedsfunktionen **i** und **a** (Eingabe und Ausgabe). Bei b) gibt es **+10 Punkte** für den korrekten Wert mit Begründung. Wird ein falscher Wert ohne Begründung angegeben, so gibt es **0 Punkte**. Ein richtiger Wert ohne Begründung gibt **+5 Punkte**, ebenso wie ein falscher Wert, der konsistent aus einer korrekten Begründung der falschen Funktionalität in a) folgt.