Prüfung B — Informatik D-MATH/D-PHYS
17. 12. 2013 13:15–14:55

Prof. 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 allgemeinen Bemerkungen gelesen und verstanden habe.

Unterschrift: .....

Allgemeine Bemerkungen und Hinweise:

1. Überprüfen Sie die Vollständigkeit der ausgeteilten Prüfungsunterlagen (drei doppelseitige Blätter mit insgesamt 5 Aufgaben und ein leeres Notizblatt)! **Tragen Sie auf dem Deckblatt gut lesbar Namen, Vornamen und Stud.-Nr. ein.**
2. Erlaubte Hilfsmittel: **Keine. Einzige Ausnahme sind Wörterbücher.**
3. Betrugsversuche führen zu sofortigem Ausschluss und können rechtliche Folgen haben.
4. **Schreiben Sie Ihre Lösungen direkt auf die Aufgabenblätter!** Pro Aufgabe ist höchstens eine gültige Version eines Lösungsversuchs zulässig. **Tipp:** Lösungsentwürfe auf separaten Blättern vorbereiten und die fertige Lösung auf die Aufgabenblätter übertragen. Falls Sie eine Lösung ändern wollen, streichen Sie den alten Lösungsversuch klar erkennbar durch. Falls auf dem Aufgabenblatt nicht mehr genug Platz für Ihre neue Lösung vorhanden ist, benutzen Sie ein separates Blatt, das mit Ihrem Namen und der Aufgabennummer beschriftet ist.
5. Wenn Sie frühzeitig abgeben möchten, übergeben Sie Ihre Unterlagen bitte einer Aufsichtsperson und verlassen Sie den Raum.
6. **Ab 14:40 Uhr kann nicht mehr frühzeitig abgegeben werden. Bleiben Sie an Ihrem Platz sitzen, bis die Prüfung beendet ist und ihre Unterlagen von einer Aufsichtsperson eingesammelt worden sind.**
7. Die Prüfung ist bestanden, wenn Sie 50 von 100 Punkten erzielen. **Viel Erfolg!**

1	2	3	4	5	$\sum$

Aufgabe 1. (18 Punkte) Geben Sie für jeden der folgenden 8 Ausdrücke Typ und Wert an! Zwischenschritte der Auswertung geben keine Punkte.Nehmen Sie den IEEE 754 Fliesskomma-Standard an.

Ausdruck	Typ	Wert
`5 + 5 / 2`		
`5.0 / 2u`		
`0.4f == 2.0/4`		
`int(2.3) - 2.0f`		
`5u * 2 / 3`		
`2 + 2012 % 2 * 3`		

Aufgabe 2. (16 Punkte) Geben Sie für jedes der vier folgenden Code-Fragmente die Folge von Zahlen an, die das Fragment ausgibt!

---

a)
```
for (int i=100; i!=0; i/=4)
  std::cout << ++i << " ";
```
Ausgabe:

---

b)
```
void f(unsigned int x) {
  std::cout << x%3 << " ";
  if (x/3 > 0) f(x/3);
}
f(34);
```
Ausgabe:

---

c)
```
for(int i=1; i<10; i+=2)
  for(int j=i; j<10; j*=2)
    std::cout << j << " ";
```
Ausgabe:

---

d)
```
double d[] = {0.6, 0.7, 0.8, 0.9};
double j = 0.5;
do {
  std::cout << (j+=d[int(j)]) << " ";
} while (j<4);
```
Ausgabe:

Aufgabe 3. (10 / 20 Punkte) Aus der Vorlesung kennen wir die *Liste* als Datenstruktur zum Speichern einer Folge von Schlüsseln des gleichen Typs. Anders als ein Feld erlaubt eine Liste effizientes Einfügen und Löschen "in der Mitte". Für den Typ `int` ist hier der Teil der Klassendefinition, der relevant für diese Aufgabe ist.

```
class List{
public:
  ...
  // POST: returns const pointer to the first node
  const ListNode* get_head() const;
  ...
private:
  ...
  ListNode* head_;
  ...
};
```

Eine Liste speichert also einen (möglicherweise Null-) Zeiger auf den Kopf-Knoten, dessen Schlüssel das erste Element der Liste ist. Ausgehend vom Kopf-Knoten können wir die Liste durchlaufen, indem wir den `get_next()`-Zeigern der Klasse `ListNode` folgen, deren relevanter Teil der Definition hier angegeben ist:

```
class ListNode {
 public:
  ...
  int get_key() const;
  ListNode* get_next() const;
  ...
 private:
  int key_;
  ListNode* next_;
};
```

a) Implementieren Sie eine Funktion die überprüft ob eine gegebene Liste aufsteigend sortiert ist. Zum Beispiel ist die Liste mit den Elementen `|1 2 3|` aufsteigend sortiert, aber die Liste mit den Elementen `|3 1 2|` ist es nicht.

b) Implementieren Sie eine Funktion die für zwei sortierte Listen als Eingabe die Elemente beider Listen in aufsteigend sortierter Reihenfolge über die Standard Ausgabe (`std::cout`) ausgibt. Zum Beispiel, für die beiden Listen `|2 4|` und `|1 3 5|` sollte die Ausgabe `1 2 3 4 5` lauten.

Sie können davon ausgehen, dass `iostream` korrekt eingebunden wurde. Allerdings dürfen Sie keine weiteren Funktionen oder Datenstrukturen aus der Standardbibliothek einbinden, z. B. auch keine Vektoren.

```
// POST: returns true if and only if the elements of l are
//       sorted in ascending order
bool is_sorted(const List& l)
{









}

// PRE: the elements in l1 and l2 are sorted in ascending order
// POST: all the elements of both lists l1 and l2 are printed in
//       ascending order to std::cout
void output_merge(const List& l1, const List& l2)
{









}
```

Aufgabe 4. (8 / 16 Punkte) Sie planen einen Urlaub nach Weihnachten. Sie haben im Internet bereits Informationen über das Reiseziel gesammelt. Unter anderem haben Sie eine Liste gefunden, in der für jeden Tag die erwarteten Sonnenstunden aufgeführt sind. Sie speichern diese Zahlen vom Typ `float` in einem Feld. Das Feld könnte zum Beispiel so aussehen:

```
float sun[] = {4.1, 3.2, 9.5, 2.1, 7.3};
```

---

a) Schreiben Sie eine Funktion, die die gesamten erwarteten Sonnenstunden in einem vorgegebenen Intervall berechnet. Die Gesamtzahl der Anweisungen darf die vorgegebene Anzahl Zeilen nicht überschreiten.

```
// PRE: [begin, end) is a valid range
// POST: returns the sum of the elements in [begin, end)
float sunshine(const float* begin, const float* end)
{
                                                    // 1
                                                    // 2
                                                    // 3
                                                    // 4
}
```

---

b) Im Urlaub wollen Sie so viel Sonne haben wie möglich. Schreiben Sie eine Funktion, die das beste Interval von `k` Tagen findet (also das Inverval der Länge `k` mit am meinsten Sonnenstunden) und die gesamte Anzahl erwarteter Sonnenstunden in diesem Interval berechnet. Zum Beispiel für $k = 2$ und das Feld `sun` wie oben, sollte die Antwort `12.7` lauten. Sie können die Funktion von Teil a) benützen. Die Gesamtzahl der Anweisungen darf die vorgegebene Anzahl Zeilen nicht überschreiten.

```
// PRE: [begin, end) is a valid range of size at least k
// POST: returns the largest sum of k consecutive elements
//       in [begin, end)
float best_holiday(const float* begin, const float* end, int k)
{
                                                    // 1
                                                    // 2
                                                    // 3
                                                    // 4
                                                    // 5
                                                    // 6
                                                    // 7
}
```

Aufgabe 5. (8 / 4 Punkte) Betrachten Sie das Lindenmayer System $\mathcal{L} = (\Sigma, P, s)$ mit Alphabet $\Sigma = \{F, +, -\}$, Startwort $s = F$, und den Produktionen

σ	$\mapsto$	$P(\sigma)$
F	$\mapsto$	$--FF-$
$+$	$\mapsto$	$+$
$-$	$\mapsto$	$-$

Ein solches System erzeugt eine unendliche Folge von Wörtern $s = w_0, w_1, \ldots$ wie folgt: Um das nächste Wort w_i vom vorhergehenden Wort w_{i-1} abzuleiten, ersetzen wir alle Symbole in w_{i-1} mit ihren Produktionen.

In unserem Beispiel führt dies zu

$$\begin{aligned} w_0 &= F \\ w_1 &= --FF- \\ w_2 &= ----FF---FF-- \end{aligned}$$

---

a) Geben Sie eine *rekursive* Definition der Funktion $t\colon \mathbb{N} \to \mathbb{N}$ mit $t(n) = |w_n|$. Das heisst, $t(n)$ ist die Länge des Wortes w_n. Sie müssen nicht argumentieren, warum Ihre Definition korrekt ist. **Hinweis**: Sie sollten mit Hilfe Ihrer Definition die folgen Werte erhalten: $t(0) = 1, t(1) = 5, t(2) = 13$.

---

b) Berechnen Sie $t(7) = |w_7|$, unter Verwendung von a) (oder irgendeiner anderen Methode)! Sie müssen nicht argumentieren, warum Ihr Wert stimmt.

$t(7) =$