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`		

Solution

Expression	Type	Value
`5 + 5 / 2`	`int`	7
`5.0 / 2u`	`double`	2.5
`0.4f == 2.0/4`	`bool`	*false*
`int(2.3) - 2.0f`	`float`	0
`5u * 2 / 3`	`unsigned int`	3
`2 + 2012 % 2 * 3`	`int`	2

Bewertung. 1 point for every 100% correct type, 2 points for every 100% correct value. No points if the answer is not 100% correct. If someone writes the literal form 1.25f instead of 1.25, this counts as 100% correct. Correct is only `unsigned int`, `int`, `float`, `double`, `bool`, no modifications like `integer`, `boolean`...

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:

Solution

a) `101 26 7 2`

b) `1 2 0 1` (base-3 representation in reverse order)

c) `1 2 4 8 3 6 5 7 9`

d) `1.1 1.8 2.5 3.3 4.2`

Bewertung. A maximum of 4 points for each of a), b), c), d) for 100% correct solutions. In each part, deduct 2 points for every wrong / missing / extra number. This means that with 3 wrong numbers, no points are left.

We deduct 1 point for writing for example 3^5 instead of 243. We deduct 1 point if there are any characters between the numbers, like commas, semicolons, etc.

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)
{









}
```

Solution

```
bool is_sorted(const List& l)
{
  const ListNode* p = l.get_head();
  while (p != 0) {
    const ListNode* next = p->get_next();
    if (next != 0 && p->get_key() > next->get_key()) return false;
    p = p->get_next();
  }
  return true;
}

void output_merge(const List& l1, const List& l2)
{
  const ListNode* p = l1.get_head();
  const ListNode* q = l2.get_head();

  while (p != 0 || q != 0) {
    if (q == 0 || (p != 0 && p->get_key() < q->get_key())) {
      std::cout << p->get_key() << " ";
      p = p->get_next();
    }
    else { //(p == 0 || (q != 0 && p->get_key() >= q->get_key()))
      std::cout << q->get_key() << " ";
      q = q->get_next();
    }
  }
  std::cout << "\n";
}
```

Bewertung.

a) The essential ingredients of the solution are

- One pointer variable to go through the list
- Single loop that increments the pointer through `get_next()`
- Comparison of corresponding keys through `get_key()`
- Correct logic to deduce output value.

No points are deducted if checking for $<$ instead of $<=$. 4 points are deducted for dereferencing 0 pointer (empty list).
4 points are deducted for not correctly advancing the pointer with `get_next`
4 points are deducted for any logic error (assuming something about elements in the list, etc.)

b) The essential ingredients of the solution are

   - Two pointer variables to go through the lists
   - Single loop that increments both pointers through `get_next()`
   - Comparison of corresponding keys through `get_key()`
   - Correct logic to deduce the order: 4 cases: first/second list ended, output element of the first/second list.

   Point guideline: award 5 points for every case that is solved correctly.

Point deductions for a) and b):

- -4 points for object mistakes, e.g., for using . instead of ->.
- -2 points for small mistakes, e.g., type mistakes.
- -4 points for not using get_head, get_key, or get_next.
- -2 points for not returning a value along some program paths.
- -4 points for any logical mistake.
- 0 points if there are no loops or recursion.

  Specifically for the second half of the exercise:

- -2 points for printing equal elements in the two lists only once.
- -5 points for any case (out of the 4) that is missing.
- no points are subtracted if more than one loops are used.

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
}
```

Solution

a)
```
float sunshine(const float* begin, const float* end)
{
  float sunshine = 0;
  while(begin != end)
    sunshine += *(begin++);
  return sunshine;
}
```

b)
```
float best_holiday(const float* begin, const float* end, int k)
{
  float best = sunshine(begin, begin + k);
  while(begin++ != end - k) {
    const float sun = sunshine(begin, begin + k);
    if (sun > best)
      best = sun;
  }
  return best;
}
```

Bewertung. Anything exceeding the # lines (hard limit!) will be awarded no points.

a) The essential ingredients of the solution are

- result variable
- loop
- adding the values (`+=` or `sum = sum + ..`)

We give 8 points for the correct logic.
4 points are deducted if the range is wrong (e.g. dereferencing the past-the-end pointer, missing the first element, empty range (`for(..; i < begin-end; ..)`), or (`for(..; *begin < *end; ..)`).
2 points are deducted for pointer mistakes: eg. `for(float i=begin; ...)` but otherwise correct logic
4 points are deducted for not correctly computing the sum, eg. `++sum`, etc.

b) The essential ingredients of the solution are

- loop to go over the range (1 time)
- compute the sum for every inverval of length k
- variable that stores the best value so far
- updating and retruning this variable

Correct solutions that do not use a) are awarded full points.
No points are deducted for assuming that the values in the array are all positive.
No points are deducted for inefficient (but correct) solutions.
4 points are deducted for wrong range; dereferencing elements outside the range 4 points are decuted for wrong range in the sum function `sum(begin, begin+k-1)`, etc. 4 points are deducted for initializing the max/min with some arbitrary value

Further points (for a) and b))were deducted as follows:

- -2 points if the return statement is wrong or missing.

- -2 points if, instead of the maximum, the minimum is computed (and vice versa).
- -2 points if there are small mistakes (not pointer mistakes).
- -2 points if there are pointer mistakes.
- -4 points of there is no variable for the best value so far.

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.

$t(7) =$

Solution

a) We observe $w_n = --w_{n-1}w_{n-1}-$ for $n \geq 1$. Hence

$$t(n) = 2t(n-1) + 3.$$

b) We can recursively apply a), or derive a closed form formula for $t(n)$:

$$t(n) = 2^{n+2} - 3.$$

Therfore, $t(7) = 509$.

Bewertung. For a), 0 points if there is no recursion. Otherwise, between 1 and 8 points, depending on the degree of correctness. If the function definition is given as C++ code, we deduce 2 points. If the base of the recurrence is missing, we also deduce 2 points. We deduce 4 points if the formula works for $t(0), t(1)$ and $t(2)$, but not for $t(7)$ (-5 points if the base case is missing). We deduce 6 points if there is a recursive formula, but everything else is wrong. Full points if the solution works for $t(0), t(1), t(2), t(7)$. Further 2 points are deducted if even the known values $t(1), t(2)$ do not correctly follow from the given formula.For b), 4 points if the solution is correct or correctly follows from a wrong *recursive* formula in a); 0 points otherwise.