Informatik 252-0847-00 Prüfung 22. 1. 2018 Lösung *B. Gärtner*

Name, Vorname: .....

Legi-Nummer: .....

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte, und dass ich die allgemeinen Richtlinien gelesen und verstanden habe.

I confirm with my signature that I was able to take this exam under regular conditions and that I have read and understood the general guidelines.

Unterschrift:

Allgemeine Richtlinien:

General guidelines:

1. Dauer der Prüfung: 60 Minuten.

   Exam duration: 60 minutes.

2. Erlaubte Unterlagen: Wörterbuch (für gesprochene Sprachen). Keine eigenen Notizblätter! Bei Bedarf stellen wir Ihnen weitere Blätter zur Verfügung.

   Permitted examination aids: dictionary (for spoken languages). No sheets of your own! We will give you extra sheets on demand.

3. Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.

   Use a pen (black or blue), not a pencil. Please write legibly. We will only consider solutions that we can read.

4. Lösungen sind direkt auf das Aufgabenblatt in die dafür vorgesehenen Boxen zu schreiben (und direkt darunter, falls mehr Platz benötigt wird). Ungültige Lösungen sind deutlich durchzustreichen! Korrekturen bei Multiple-Choice Aufgaben bitte unmissverständlich anbringen! Lösungen auf Notizblättern werden nur in Notfällen berücksichtigt. Bitte kontaktieren Sie in diesem Fall eine Aufsichtsperson.

   Solutions must be written directly onto the exam sheets in the provided boxes (and directly below, if more space is needed). Invalid solutions need to be crossed out clearly. Provide corrections to answers of multiple choice questions without any ambiguity! Solutions on extra sheets will be considered only in emergencies. In this case, please contact a supervisor.

5. Es gibt keine Negativpunkte für falsche Antworten.

   There are no negative points for false answers.

6. Störungen durch irgendjemanden oder irgendetwas melden Sie bitte sofort der Aufsichtsperson.

   If you feel disturbed by anyone or anything, let the supervisor of the exam know immediately.

7. Wir sammeln die Prüfung zum Schluss ein. Wichtig: stellen Sie unbedingt selbst sicher, dass Ihre Prüfung von einem Assistenten eingezogen wird. Stecken Sie keine Prüfung ein und lassen Sie Ihre Prüfung nicht einfach am Platz liegen. Dasselbe gilt, wenn Sie früher abgeben wollen: bitte melden Sie sich lautlos, und wir holen die Prüfung ab. Vorzeitige Abgaben sind nur bis 15 Minuten vor Prüfungsende möglich.

   We collect the exams at the end. Important: you must ensure that your exam has been collected by an assistant. Do not take any exam with you and do not leave your exam behind on your desk. The same applies when you want to finish early: please contact us silently and we will collect the exam. Handing in your exam ahead of time is only possible until 15 minutes before the exam ends.

8. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen.

   If you need to go to the toilet, raise your hand and wait for a supervisor.

9. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

   We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.

Aufgabe	1	2	3	4	5	6	7	8	$\sum$	Bonus
Punkte										
Maximum	9	6	6	6	6	8	7	12	60	0.25

Viel Erfolg! *Good luck!*

Aufgaben / *Tasks*

1 Typen und Werte / Types and values *(9 Punkte)*

Geben Sie für jeden der sechs Ausdrücke unten jeweils C++-Typ (0.5 P) und Wert (1 P) an! Nehmen Sie für Fliesskommazahlen den Standard IEEE 754 an! *For each of the six expressions below, provide the C++ type (0.5 P) and value (1 P)! For floating point numbers, assume the IEEE 754 standard!*

(a) `8 + 5 / 3` — 1.5 P

Typ/*Type*	Wert/*Value*
`int`	`9`

(b) `1.0 * 0.1 == 0.1 * 1.0` — 1.5 P

Typ/*Type*	Wert/*Value*
`bool`	`true`

(c) `1e1 * 2e2` — 1.5 P

Typ/*Type*	Wert/*Value*
`double`	`2000`

(d) `5 / 2.0f + 5 / 2.0` — 1.5 P

Typ/*Type*	Wert/*Value*
`double`	`5`

(e) `!false || true && false` — 1.5 P

Typ/*Type*	Wert/*Value*
`bool`	`true`

(f) `17 % 5u` — 1.5 P

Typ/*Type*	Wert/*Value*
`unsigned int`	`2`

2 Schleifenausgaben / Loop outputs *(6 Punkte)*

Geben Sie für jedes der folgenden drei Code-Stücke die Ausgabe an, die generiert wird! Nehmen Sie für Fliesskommazahlen den Standard IEEE 754 an! Falls es sich um eine Endlosschleife handelt, schreiben Sie "Endlosschleife"!

For each of the following three code snippets, provide the output that is being generated! For floating point numbers, assume the IEEE 754 standard! In case of an infinite loop, write "infinite loop"!

(a)

```
for (double d = 0.25; d != 16; d = 2*(d+1))
   std::cout << d << ", ";
```

2 P

Ausgabe/*Output*:

```
0.25, 2.5, 7,
```

(b)

```
int a[] = {1, 2, -2, -1};
int* p = a;
do {
  std::cout << *p << ", ";
  p += *p;
} while (p != a);
```

Ausgabe/*Output*:

```
1, 2, -1, -2,
```

2 P

(c)

```
unsigned int n = 12;
while (n > 1) {
  std::cout << n << ", ";
  if (n % 2 == 0)
    n = n / 2;
  else
    n = 2 * n;
}
```

Ausgabe/*Output*:

```
infinite loop
```

2 P

3 Inverse modulo p / Inverses modulo p *(6 Punkte)*

Betrachten Sie das folgende Programmfragment, das eine Funktion zur Berechnung inverser Elemente modulo einer Primzahl p definiert. Das inverse Element von $a \in \{1, \ldots, p-1\}$ modulo p ist die eindeutige Zahl $x \in \{1, \ldots, p-1\}$ mit $ax \equiv 1 \mod p$. Ihre Aufgabe ist es, die Funktion `inverse` (spezifiziert durch Vor- und Nachbedingung) zu implementieren. Sie müssen die Vorbedingung dabei nicht prüfen. Die Tabelle unten zeigt die erwarteten Ergebnisse modulo 7.

a	inverse(a,7)	a * inverse(a,7)
1	1	1 $(\equiv 1 \mod 7)$
2	4	8 $(\equiv 1 \mod 7)$
3	5	15 $(\equiv 1 \mod 7)$
4	2	8 $(\equiv 1 \mod 7)$
5	3	15 $(\equiv 1 \mod 7)$
6	6	36 $(\equiv 1 \mod 7)$

Consider the following program fragment that defines a function for computing inverse elements modulo a prime number p. The inverse element of $a \in \{1, \ldots, p-1\}$ modulo p is the unique number $x \in \{1, \ldots, p-1\}$ such that $ax \equiv 1 \mod p$. Your task is to implement the function `inverse` (specified by pre- and postcondition). In doing so, you don't have to check the precondition.The table above shows the expected results modulo 7.

```
// PRE: p is prime, 0 < a < p
// POST: computes the inverse of a modulo p, i.e. the unique number
//       x, 0 < x < p such that a * x is congruent to 1 modulo p
unsigned int inverse (unsigned int a, unsigned int p)
{
```

```
1.  unsigned int x = 1;
2.  while (a * x % p != 1) {
3.      x += 1;
4.  }
5.  return x;
```

```
}
```

4 Vor-/Nachbedingungen / Pre-/Postconditions *(6 Punkte)*

Analysieren Sie folgende Funktionen und geben Sie jeweils eine Vorbedingung (1P) und eine Nachbedingung (2P) an, so dass das Verhalten der Funktion komplett spezifiziert ist!

```
// PRE:
```

```
[begin,end) is a valid range
```

```
// POST:
```

```
The order of elements in [begin,end) is reversed and true is
returned if and only if the range has even length
```

```
bool f (int* begin, int* end)
{
  while (begin < end)
    std::swap (*(begin++), *(--end));
  return begin == end;
}
```

```
// PRE:
```

```
n > 0
```

```
//POST:
```

```
returns n
```

```
unsigned int g (unsigned int n) {
  if (n == 1)
    return 1;
  else
    return 2 * g (n / 2) + n % 2;
}
```

Analyse the functions above and in both cases provide a precondition (1P) and a postcondition (2P) so that the behavior of the function is completely specified!

5 Binärdarstellung / Binary representation *(6 Punkte)*

Berechnen Sie die Binärdarstellungen $b_0.b_{-1}b_{-2}\cdots$ der unten angegebenen rationalen Zahlen $q \in (0,2)$, das heisst, schreiben Sie q jeweils in der Form

$$q = \sum_{i=0}^{\infty} b_{-i}2^{-i}, \quad b_{-i} \in \{0,1\} \text{ für alle } i.$$

Falls die Darstellung nicht endlich ist, geben Sie die exakte Periode an! Im folgenden finden Sie drei Beispiele.

q	$b_0.b_{-1}b_{-2}\cdots$
$3/2$	1.1
$7/8$	0.111
$4/3$	$1.\overline{01}$

Compute the binary representations $b_0.b_{-1}b_{-2}\cdots$ of the rational numbers $q \in (0,2)$ given below, i.e. write q in the form

$$q = \sum_{i=0}^{\infty} b_{-i}2^{-i}, \quad b_{-i} \in \{0,1\} \text{ for all } i.$$

If the representation is not finite, provide the exact period! Above, you find three examples.

q	$b_0.b_{-1}b_{-2}\cdots$
$5/4$	1.01
$31/32$	0.11111
$4/7$	$0.\overline{100}$
$3/7$	$0.\overline{011}$

6 EBNF *(8 Punkte)*

Die folgende EBNF beschreibt eine "Programmierspache" zur Erzeugung von Turtlegrafik-Befehlsfolgen. Geben Sie für jede Aussage auf dieser und der nächsten Seite an, ob sie zutrifft oder nicht.

Anmerkung: Leerschläge sind im Rahmen der EBNF bedeutungslos.

```
program = command | repetition | "(" { program } ")"
repetition = "REPEAT" number program
number = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
command = "F" | "+" | "-"
```

The EBNF above describes a "programming language" for the generation of turtle graphic command sequences. For each statement on this and the next page, decide whether it is true or not.

Remark: *Whitespaces are irrelevant in the context of this EBNF.*

(a) **W**ahr oder **f**alsch? ***t**rue or **f**alse?* `false` 1 P

Die folgende Zeichenkette ist ein gültiges Programm (`program`) nach der EBNF:
The following string is a valid program (`program`) according to the EBNF:

```
F+F-
```

(b) **W**ahr oder **f**alsch? ***t**rue or **f**alse?* `true` 1 P

Die folgende Zeichenkette ist ein gültiges Programm (`program`) nach der EBNF:
The following string is a valid program (`program`) according to the EBNF:

```
REPEAT 4 (F+)
```

(c) **W**ahr oder **f**alsch? ***t**rue or **f**alse?* `true` 1 P

Die folgende Zeichenkette ist ein gültiges Programm (`program`) nach der EBNF:
The following string is a valid program (`program`) according to the EBNF:

```
REPEAT 4 REPEAT 5 F
```

(d) **W**ahr oder **f**alsch? *true or false?* `true` 1 P

Die folgende Zeichenkette ist ein gültiges Programm (`program`) nach der EBNF:
The following string is a valid program (`program`) according to the EBNF:
`REPEAT 1 ()`

(e) **W**ahr oder **f**alsch? *true or false?* `true` 1 P

Die folgende Zeichenkette ist ein gültiges Programm (`program`) nach der EBNF:
The following string is a valid program (`program`) according to the EBNF:
`(REPEAT 9 F + REPEAT 9 F)`

(f) **W**ahr oder **f**alsch? *true or false?* `false` 1 P

Die folgende Zeichenkette ist ein gültiges Programm (`program`) nach der EBNF:
The following string is a valid program (`program`) according to the EBNF:
`REPEAT 2`

(g) **W**ahr oder **f**alsch? *true or false?* `false` 1 P

Die folgende Zeichenkette ist ein gültiges Programm (`program`) nach der EBNF:
The following string is a valid program (`program`) according to the EBNF:
`(- REPEAT 10 F )`

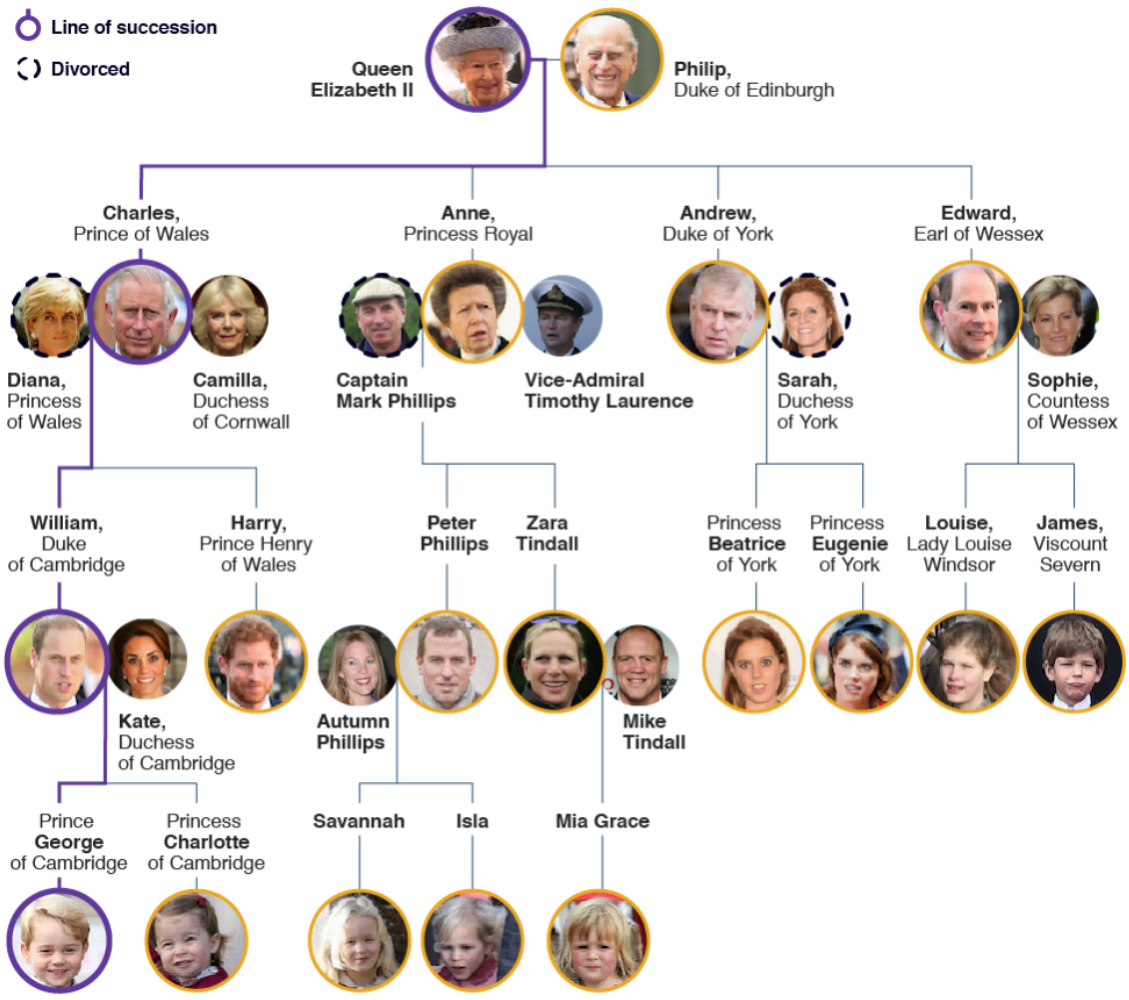
(h) **W**ahr oder **f**alsch? *true or false?* `true` 1 P

Die folgende Zeichenkette ist ein gültiges Programm (`program`) nach der EBNF:
The following string is a valid program (`program`) according to the EBNF:
`(REPEAT 5 FF)`

7 Stammbaum / Family tree *(7 Punkte)*

Die Klasse `person` auf der nächsten Seite repräsentiert eine Person durch ihren Namen und einen (möglicherweise leeren) Vektor von Zeigern auf ihre Kinder (nur Member, die für diese Aufgabe relevant sind, werden angegeben). Mit Hilfe dieser Klasse können Stammbäume wie der im Bild angegebene aufgebaut werden.

Ergänzen Sie die Memberfunktion `number_of_children (unsigned int g)`, so dass sie für eine gegebene Person korrekt berechnet, wieviele Kinder vom Grad g sie hat. Dabei sind die Kinder vom Grad 0 die eigenen Kinder, die Kinder vom Grad 1 die Enkel, die Kinder vom Grad 2 die Urenkel usw. Im Beispiel hat Elizabeth 4 Kinder vom Grad 0, 8 vom Grad 1, 5 vom Grad 2 und (noch) keine von grösserem Grad.



`http://www.bbc.com/news/uk-23272491`

The class `person` on this page represents a person by its name and a (possibly empty) vector of pointers to its children (only members relevant for this task are provided). With the help of this class, we can build family trees such as the one in the picture.

Complete the member function `number_of_children (unsigned int g)`, such that for a given person, it correctly computes the number of children of grandness g. Children of grandness 0 are the own children, children of grandness 1 are the grand children, children of grandness 2 are the grand grand children etc. In the example, Elizabeth has 4 children of grandness 0, 8 of grandness 1, 5 of grandness 2, and none of higher grandness (yet).

```
struct person {
  std::string name;
  std::vector<person*> children;

  // returns the number of children of *this
  unsigned int number_of_children() const
  {
    return children.size();
  }

  // returns the number of children of grandness g of *this
  //   g = 0: children
  //   g = 1: grand children
  //   g = 2: grand grand children ...
  unsigned int number_of_children (unsigned int g)
  {
    if (g == 0)
      return number_of_children();
    else {
     unsigned int c = 0;
     for (int i = 0; i < children.size(); ++i)
       c += children[i]->number_of_children (g-1);
     return c;
    }
  }
};
```

8 Unimodale Folgen / Unimodal sequences *(12 Punkte)*

Eine nichtleere Folge $(x_1, \ldots, x_n)$ von Zahlen heisst unimodal, falls es einen Index i gibt, so dass $x_1 < \ldots < x_{i-1} < x_i > x_{i+1} > \ldots > x_n$. Das heisst, die Folge ist strikt monoton steigend bis zum i-ten Element und danach strikt monoton fallend. Zum Beispiel ist die Folge $(1, 2, 3, 2, 1)$ unimodal mit $i = 3$. Jede strikt monton steigende Folge ist unimodal (in diesem Fall ist $i = n$), ebenso wie jede strikt monoton fallende Folge ($i = 1$). Folgen der Länge 1 sind immer unimodal ($i = 1 = n$). Die Folgen $(1, 2, 3, 2, 1, 2)$ und $(2, 2)$ sind nicht unimodal.

Implementieren Sie die Funktion `is_unimodal` unten so, dass sie korrekt testet, ob die Folge, die durch den nichtleeren Bereich `[begin, end)` repräsentiert wird, unimodal ist.

```
// PRE: [begin, end) is a valid nonempty range
// POST: returns true if and only if [begin, end) is unimodal
bool is_unimodal (const int* begin, const int* end)
```

```
1.  int x = *begin;
2.  const int* p = begin+1;
3.  while (p < end && x < *p) {
4.    x = *p;
5.    ++p;
6.  }
7.  while (p < end && x > *p) {
8.    x = *p;
9.    ++p;
hom10.  }
11.  return (p == end);
```

```
}
```

A nonempty sequence $(x_1, \ldots, x_n)$ of numbers is unimodal, if there is an index i such that $x_1 < \ldots < x_{i-1} < x_i > x_{i+1} > \ldots > x_n$. This means, the sequence is strictly monotone increasing up to the i-th element, and strictly monotone decreasing afterwards. For example, the sequence $(1, 2, 3, 2, 1)$ is unimodal with $i = 3$. Every strictly monotone increasing sequence is unimodal (in this case, $i = n$), and so is every strictly monotone decreasing sequence ($i = 1$). Sequences of length 1 are always unimodal ($i = 1 = n$). The sequences $(1, 2, 3, 2, 1, 2)$ and $(2, 2)$ are not unimodal.

Implement a function `is_unimodal` that tests whether the sequence represented by the nonempty range `[begin, end)` is unimodal.