Informatik 252-0847-00 Prüfung 12. 8. 2016 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:

1. Dauer der Prüfung: 60 Minuten.
2. Erlaubte Unterlagen: Wörterbuch (für gesprochene Sprachen). Keine eigenen Notizblätter! Bei Bedarf stellen wir Ihnen weitere Blätter zur Verfügung.
3. Benützen Sie einen Kugelschreiber (blau oder schwarz) und keinen Bleistift. Bitte schreiben Sie leserlich. Nur lesbare Resultate werden bewertet.
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 nicht berücksichtigt.
5. Es gibt keine Negativpunkte für falsche Antworten.
6. Störungen durch irgendjemanden oder irgendetwas melden Sie bitte sofort der Aufsichtsperson.
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.
8. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen.
9. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

General guidelines:

Exam duration: 60 minutes.

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

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

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 not be considered.

There are no negative points for false answers.

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

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.

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

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	$\sum$
Punkte									
Maximum	6	7.5	7.5	8	8	6	6	12	61

Viel Erfolg! *Good luck!*

Aufgaben / *Tasks*

1 C++-Allgemeinwissen *(6 Punkte)*

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

Jeder Ausdruck hat einen Typ. *Every expression has a type.*

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

Call by value erlaubt Funktionen, die Werte ihrer Aufrufargumente zu ändern. *Call by value enables functions to change the values of their call arguments.*

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

Es gibt Compiler, die erkennen, ob jedes Programm frei von Endlosschleifen ist. *There are compilers that detect whether every program is free from infinite loops.*

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

Einige Berechnungsprobleme können in C++ nur mit Hilfe von Rekursion gelöst werden. *Some computation problems can only be solved in C++ with the help of recursion.*

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

Jede Klasse muss einen Default-Konstruktor haben. *Every class must have a default constructor.*

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

Private Datenmember können nur in privaten Memberfunktionen benutzt werden. *Private data members can only be used in private member functions.*

2 Typen und Werte (Fundamentale Typen) *(7.5 Punkte)*

Geben Sie für jeden der fünf Ausdrücke auf der rechten Seite jeweils C++-Typ (0.5 P) und Wert (1 P) an! Wenn der Wert nicht definiert ist, schreiben Sie "undefiniert".

Die verwendeten Variablen sind wie folgt deklariert und initialisiert.

```
int n = 9;
int f = 5;
int d = 2;
unsigned int u = 8;
bool a = false;
bool b = true;
```

For each of the five expressions on the right, provide the C++ type (0.5 P) and value (1 P)! If a value is undefined, write "undefined".

The used variables have been declared and initialized as shown above.

(a) `n / f * d` — 1.5 P

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

(b) `-n - ++f` — 1.5 P

Typ/ *Type*	Wert/ *Value*
int	-15

(c) `0x11 * 0xa` — 1.5 P

Typ/ *Type*	Wert/ *Value*
int	170

(d) `u-7 == -1` — 1.5 P

Typ/ *Type*	Wert/ *Value*
bool	false

(e) `a || !b && !a || b` — 1.5 P

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

0.5 points for every correct type, 1 point for every correct value

3 Typen und Werte (Felder und Zeiger) *(7.5 Punkte)*

Geben Sie für jeden der fünf Ausdrücke auf der rechten Seite jeweils C++-Typ (0.5 P) und Wert (1 P) an! Wenn der Wert nicht definiert ist, schreiben Sie "undefiniert".

Die verwendeten Variablen sind wie folgt deklariert und initialisiert.

```
int a[2] = {1,2};
int* x = &a[0];
int* y = a+1;
double d[2][2] = {{4,3},{2,1}};
```

For each of the five expressions on the right, provide the C++ type (0.5 P) and value (1 P)! If a value is undefined, write "undefined".

The used variables have been declared and initialized as shown above.

(a) `a[y-x]` — 1.5 P

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

(b) `a[*x] * x[*a]` — 1.5 P

Typ/ *Type*	Wert/ *Value*
`int`	`4`

(c) `d[*y][*y]` — 1.5 P

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

(d) `d[0][1] * d[1][0]` — 1.5 P

Typ/ *Type*	Wert/ *Value*
`double`	`6`

(e) `&d[1][1] < &d[1][0]` — 1.5 P

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

0.5 points for every correct type, 1 point for every correct value

4 Programmausgaben *(8 Punkte)*

Betrachten Sie folgendes Programm. Beantworten Sie die Fragen auf der rechten Seite.

```
#include <iostream>

void func_a (char* start, char* end) {
   for (char* it = start; it < end; ++it)
      *it = *it-1;
}

void func_b (char* start, char* end) {
   for (const char* it = start; it < end; it+=2)
      std::cout << *it << " ";
}

void func_c (int b) {
   if (b > 0) {
      func_c (b/2);
      if (b % 2 == 1)
         std::cout << "X";
      else
         std::cout << "U";
   }
}

bool func_d (int* a, int& i) {
   i = a[i];
   return i>0;
}

int main() {
  // insert code from right hand side here
  return 0;
}
```

Consider the program above. Answer the questions on the right hand side!

Was gibt das Programm auf der linken Seite jeweils aus, wenn man folgende Codestücke in die `main`-Funktion einfügt?

What is the respective output of the program on the left hand side when the following code pieces are inserted into the main function?

(a) 2 P

```
unsigned int x = 4;
do {
   std::cout << x << " ";
   x = (2 * x + 3) % 17;
} while (x != 0);
```

```
4 11 8 2 7
```

(b) 2 P

```
char src[] = { 'd', 'c', 's', 'b', 'h', 'l'};
func_a (src, src+3);
func_b (src, src+6);
```

```
c r h
```

(c) 2 P

```
func_c (22);
```

```
XUXXU
```

(d) 2 P

```
int i = 0;
int test[] = {2,4,1,0,3};
while (func_d (test, i))
  std::cout << test[i];
```

```
1430
```

5 Normalisierte Fliesskommazahlen *(8 Punkte)*

Wir betrachten das unten angegebene normalisierte Fliesskommazahlensystem F^*. Beantworten Sie die Fragen auf der rechten Seite!

Anmerkung: Falls nötig, runden Sie wie folgt: bei einer 1 direkt hinter der letzten signifikanten Stelle wird aufgerundet, bei einer 0 wird abgerundet.

Beispiel: in F^* wird die binär dargestellte Zahl $1.01\underline{0}..$ zu 1.01 abgerundet, während $1.01\underline{1}..$ zu 1.10 aufgerundet wird.

$F^*\left(\beta, p, e_{\min}, e_{\max}\right)$ mit / *with*

$\beta = 2$

$p = 3$

$e_{\min} = -3$

$e_{\max} = 3$

Consider the normalized floating point number system F^* as defined above. Answer the questions on the right hand side!

***Remark:** If necessary, use the following rounding mode: if there is a 1 directly behind the last significant digit, round up, and if there is a 0, round down.*

***Example:** in F^*, the binary represented number $1.01\underline{0}..$ is rounded down to 1.01, while $1.01\underline{1}..$ is rounded up to 1.10.*

(a) Wie viele unterschiedliche positive Werte enthält das normalisierte Fliesskommazahlensystem F^*? 1 P

How many different positive values does the normalized floating point number system F^* contain?

28

(b) Geben Sie die grösste Zahl und die kleinste positive Zahl an, die im normalisierten Fliesskommazahlensystem F^* enthalten ist. Beide Antworten sind in **dezimaler Darstellung** anzugeben. 2 P

*Provide the largest number and the smallest positive number contained in the normalized floating point number system F^*. Provide both answers in **decimal representation**.*

grösste Zahl / *largest number*	14
kleinste positive Zahl / *smallest positive number*	1/8

(c) Berechnen Sie folgende Ausdrücke so wie sie geklammert sind, indem Sie gemäss den Regeln für das Rechnen mit Fliesskommazahlen jedes Zwischenresultat (und das Endresultat) im gegebenen Fliesskommazahlensystem darstellen. Vergessen Sie das korrekte Runden nicht (siehe Anmerkung auf der linken Seite)! 5 P

Compute the following expressions as the parentheses suggest, representing each intermediate result (and the final result) in the given normalized floating point number system according to the rules of computing with floating point numbers. Do not forget to round correctly (see Remark on the left hand side)!

$(0.75 + 0.5) + 10$

	dezimal *decimal*	binär *binary*
	0.75	$1.10 \cdot 2^{-1}$
+	0.5	$1.00 \cdot 2^{-1}$
=		$1.01 \cdot 2^{0}$
+	10	$1.01 \cdot 2^{3}$
=	12 ←	$1.10 \cdot 2^{3}$

$(10 + 0.75) + 0.5$

	dezimal *decimal*	binär / *binary*
	10	$1.01 \cdot 2^{3}$
+	0.75	$1.10 \cdot 2^{-1}$
=		$1.01 \cdot 2^{3}$
+	0.5	$1.00 \cdot 2^{-1}$
=	10 ←	$1.01 \cdot 2^{3}$

6 BNF I *(6 Punkte)*

Die folgende BNF definiert eine Sprache zur Beschreibung von Präfix-Bäumen. Mit diesen können Mengen von Wörtern effizient gespeichert werden, indem gemeinsame Wortanfänge geteilt werden. Ein geklammerter Ausdruck steht für einen Präfix-Baum, die Zeichen sind Kantenbeschriftungen. Ein illustrierendes Beispiel ist ganz unten auf der Seite zu finden (**fett:** die im Präfix-Baum gespeicherten Wörter; jedes Blatt entspricht einem gespeicherten Wort).

Beantworten Sie die Fragen auf der rechten Seite!

Anmerkung: Leerschläge sind im Rahmen der BNF bedeutungslos.

```
Tree = '(' Branches ')'
Branches = Branch | Branch Branches
Branch = Label | Label Tree
Label = 'a' | 'b' | ... | 'z'
```

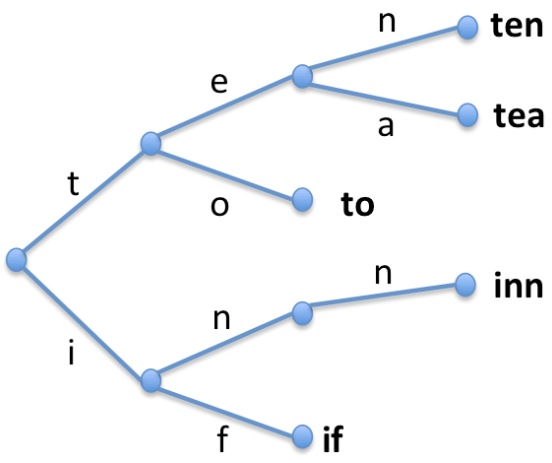
*The BNF displayed above defines a language for the description of prefix trees. These can be used to efficiently store a set of words by sharing common beginnings. An expression in parentheses stands for a prefix tree, characters designate edge labels. An illustrating example is shown at the bottom of the page (**bold:** the words stored in the leaves of the prefix tree; every leaf of the tree corresponds to a stored word).*

Answer the questions on the right hand side!

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

Beispiel / *Example*:

(t (e (n a) o) i (n (n) f)) ⇔



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

Die folgende Zeichenkette ist ein gültiger Präfix-Baum (`Tree`) nach der BNF:
The following string is a valid prefix tree (`Tree`) according to the BNF:

```
a
```

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

Die folgende Zeichenkette ist ein gültiger Präfix-Baum (`Tree`) nach der BNF:
The following string is a valid prefix tree (`Tree`) according to the BNF:

```
(a b c)
```

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

Die folgende Zeichenkette ist ein gültiger Präfix-Baum (`Tree`) nach der BNF:
The following string is a valid prefix tree (`Tree`) according to the BNF:

```
(a b (c d e) f g (h i j) k)
```

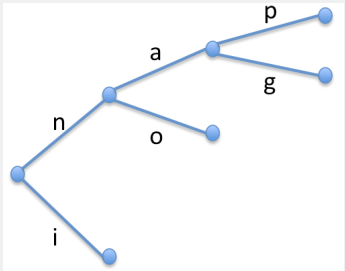
(d) **W**ahr oder **f**alsch? *true or false?* `false` 1 P

Die folgende Zeichenkette ist ein gültiger Präfix-Baum (`Tree`) nach der BNF:
The following string is a valid prefix tree (`Tree`) according to the BNF:

```
(a b) (c d)
```

(e) Zeichnen Sie den Präfix-Baum, welcher zu folgender Zeichenkette gehört. 2 P
Draw the prefix tree that belongs to the following string.

(n (a (p g) o) i)



7 BNF II *(6 Punkte)*

Betrachten Sie folgenden Code und ergänzen Sie ihn anhand der Beschreibung auf der rechten Seite mit den vier fehlenden Ausdrücken. *Consider the following code and complete it with the four missing expressions, according to the description on the right hand side.*

```
// POST: leading whitespace characters are extracted from is, and the
//     first non-whitespace character is returned (0 if there is none)
char lookahead (std::istream& is);

// PRE: Tree = '(' Branches ')'
// POST: Extracts tree from is and returns its depth
int Tree (std::istream& is) {
    char c; is >> c; // extract '('
    const int depth = [Branches (is)];
    is >> c; // extract ')'
    return depth;
}
// PRE: Branches = Branch | Branch Branches
// POST: Extracts all branches from is and returns maximum depth of a branch
int Branches (std::istream& is) {
    int depth = Branch (is); // extract branch and get its depth
    if (lookahead (is) != ')') {
        const int bDepth = Branches(is);
        if ([bDepth > depth])
          [depth = bDepth];
    }
    return depth;
}
// PRE: Branch = Label | Label Tree
// POST: Extracts single branch from is and returns its depth
int Branch (std::istream& is) {
   char c; is >> c; // extract label
   if (lookahead (is) == '(')
       return [1 + Tree (is)];
   return 1;
}
```

Die folgende main-Funktion soll zu einem gültigen Präfix-Baum, welcher gemäss der BNF der vorherigen Aufgabe am Eingabestrom vorliegt, die Tiefe (depth) berechnen; die Tiefe eines Baums ist die Länge des längsten Pfades von der Wurzel bis zu einem Blatt; in einem Präfix-Baum entspricht die Tiefe der Länge des längsten gespeicherten Wortes. Im Beispiel auf Seite 12 ist die Tiefe 3, und dies entspricht der Länge der längsten gespeicherten Wörter **ten**, **tea** und **inn**.

*The following main function shall return the depth of a valid prefix tree that is provided at the input stream according to the BNF defined in the previous task. The depth of a tree is the length of the longest path from the root to a leaf. In a prefix tree, the depth corresponds to the length of the longest stored word. In the example on page 12, the depth is 3, and this corresponds to the length of the longest stored words **ten**, **tea**, and **inn**.*

```
int main() {
  const int depth = Tree(std::cin);
  std::cout << "Longest stored word has length " << depth << "\n";
  return 0;
}
```

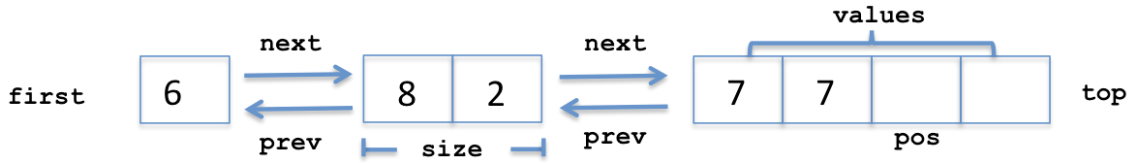
(a) Vervollständigen Sie den Code der Funktion `Tree` entsprechend. 2 P
Complete the code of the function `Tree` accordingly.

(b) Vervollständigen Sie den Code der Funktion `Branches` entsprechend. 2 P
Complete the code of the function `Branches` accordingly.

(c) Vervollständigen Sie den Code der Funktion `Branch` entsprechend. 2 P
Complete the code of the function `Branch` accordingly.

8 Verdoppelnder Stapel *(12 Punkte)*

Die Klasse `DoublingStack` verwaltet einen Stapel als doppelt verkettete Liste von Feldern, wobei jedes Feld doppelt so gross ist wie das vorhergehende. Dadurch muss nicht bei jeder `push`-Operation neuer Speicher angefordert werden, sondern nur, wenn das jeweils oberste Feld voll ist. Die folgende Skizze veranschaulicht einen verdoppelnden Stapel nach Pushen der fünf Elemente $6, 8, 2, 7, 7$. Bearbeiten Sie die Aufgaben auf der rechten Seite.



The class `DoublingStack` maintains a stack as doubly-linked list of arrays where each array has twice the size of the previous one. This ensures that new memory does not need to be allocated in every `push` operation, but only if the respective top array is full. The sketch above visualizes a doubling stack after pushing the five elements $6, 8, 2, 7, 7$. Execute the tasks on the right hand side.

```
struct Node { // for one array in the list
  int* values; int size; Node* next; Node* prev; // data members

  Node (int s, Node* p)
   :  values (new int[s]), size (s), next (0), prev (p) // constructor
  {}
  ~Node() { delete[] values; } // destructor
};

class DoublingStack {
  Node* first; Node* top;  int pos; // data members
public:
  DoublingStack ()
  : first (new Node (1, 0)), top (first), pos (0) // default constructor
  {}
  // POST: puts value on the stack
  void push (int value) {
    if (pos == top->size) { // top array full
      top->next = new Node (2 * top->size, top);
      top = top->next; pos = 0;
    }
    top->values[pos++] = value;
  }
...
};
```

(a) Wieviele Felder enthält ein initial leerer `DoublingStack`, nachdem seine Funktion `push` N mal aufgerufen wurde? 2 P

How many arrays does an initially empty `DoublingStack` contain after its function `push` has been called N times?

N = 7: 3 N = 33: 6

(b) Implementieren Sie den Destruktor `~DoublingStack()`, so dass er den angeforderten Speicher wieder freigibt: 4 P

Implement the destructor `~DoublingStack()` such that it frees all allocated memory:

```
~DoublingStack() {
    Node* node = first;
    while (node != 0 ) {
        Node* tmp = node ;
        node = node->next ;
        delete tmp ;
    }
}
```

(c) Implementieren Sie die Funktion `pop` der Klasse `DoublingStack`, welche zuerst das `top`-Feld löscht, wenn es leer ist, dann das oberste Element vom Stapel entfernt und seinen Wert zurückgibt. 6 P

Implement the function `pop` of the class `DoublingStack` which first deletes the `top` array if it is empty, then removes the top element from the stack and returns its value.

```
// PRE: *this is not empty
int pop() {
   if (pos == 0 ) {
      top = top->prev ;
      pos = top->size ;
      delete top->next ;
      top->next = 0 ;
   }
   return top->values[--pos] ;
}
```