Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Prüfung **(Lösung)**

Informatik I (D-ITET)

Felix Friedrich, Martin Bättig

ETH Zürich, 23.8.2017.

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).
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 bitte deutlich durchstreichen! Korrekturen bei Multiple-Choice Aufgaben unmissverständlich anbringen!
5. Falls Sie sich durch irgendjemanden oder irgendetwas gestört fühlen, melden Sie dies sofort der Aufsichtsperson.
6. 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.
7. Wenn Sie zur Toilette müssen, melden Sie dies einer Aufsichtsperson durch Handzeichen. Es darf zur gleichen Zeit immer nur eine Studentin oder ein Student zur Toilette.
8. Wir beantworten keine inhaltlichen Fragen während der Prüfung. Kommentare zur Aufgabe schreiben Sie bitte auf das Aufgabenblatt.

General guidelines:

1. *Exam duration: 60 minutes.*
2. *Permitted examination aids: dictionary (for spoken languages).*
3. *Use a pen (black or blue), not a pencil. Please write legibly. We will only correct solutions that we can read.*
4. *All solutions must be written directly onto the exercise sheets in the provided boxes (and directly below, if more space is needed). Invalid solutions need to be crossed out clearly. Corrections to answers of multiple choice questions must be provided without any ambiguity.*
5. *If you feel disturbed by anyone or anything, immediately let the supervisor of the exam know this.*
6. *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 preliminarily is only possible until 15 minutes before the exam ends.*
7. *If you need to go to the toilet, raise your hand and wait for a supervisor. Only one student can go to the toilet at a time.*
8. *We will not answer any content-related questions during the exam. Please write comments referring to the tasks on the exam sheets.*

Question:	1	2	3	4	5	6	7	Total
Points:	6	8	8	6	8	8	7	51
Score:								

Aufgabe 1: Typen und Werte (Basistypen) (6P)

Geben Sie für jeden der Ausdrücke auf der nächsten Seite jeweils C++Typ und Wert an. Wenn der Wert nicht bestimmt werden kann, schreiben Sie "undefiniert".
Die verwendeten Variablen sind wie folgt deklariert / initialisiert.

For each of the expressions on the next page provide the C++type and value. If a value cannot be determined, write "undefined".
The used variables are declared as follows.

```
int i = 10;
double d = 2;
int m = 7;
int pre = 1;
int post = 1;
unsigned int u = 0;
```

(a) `1 / 4 * d` /1P

Typ / *Type*	Wert / *Value*
`double`	`0`

(b) `m % m` /1P

Typ / *Type*	Wert / *Value*
`int`	`0`

(c) `i = 10` /1P

Typ / *Type*	Wert / *Value*
`int`	`10`

(d) `i += i` /1P

Typ / *Type*	Wert / *Value*
`int`	`20`

(e) `++pre / post--` /1P

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

(f) `u-5 < 6` /1P

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

Aufgabe 2: Konstrukte (8P)

Geben Sie zu folgenden Codestücken jeweils die erzeugte Ausgabe an.

Provide the output for each of the following pieces of code.

/2P (a)

```
int a[] = {1,2,3};
int* b = a;
std::cout << *(b+*(b+1));
```

Ausgabe / *Output*: 3

/2P (b)

```
int i = 3;
int& j = i;
++j;
std::cout << j + i;
```

Ausgabe / *Output*: 8

/2P (c)

```
char s[] = "DTI";
char* n = s;
int t[] = {1,0,-1};
int* i = t;
*(n++) += *(i++);
*(++n) += *(++i);
std::cout << s;
```

Ausgabe / *Output*: ETH

/2P (d)

```
struct S{
    S* n;
    int v;
    S(int V, S* N): n(N), v(V) {};
};

S s = S(10, new S(20, &s));
std::cout << (s.n->n->v);
```

Ausgabe / *Output*: 10

Aufgabe 3: Zahlendarstellungen (8P)

(a) Geben Sie ein möglichst knappes normalisiertes Fliesskommazahlensystem an, mit welchem sich die folgenden dezimalen Werte gerade noch genau darstellen lassen: jede Verkleinerung von p, $e_{\max}$ oder $-e_{\min}$ muss dazu führen, dass mindestens eine Zahl nicht mehr dargestellt werden kann.
Hinweis: p zählt auch die führende Ziffer.
Tipp: Schreiben Sie sich die normalisierte Binärzahldarstellung der Werte auf, wenn sie für Sie nicht offensichtlich ist.

Provide a smallest possible normalized floating point number system that can still represent the following values exactly: any decrease of the numbers p, $e_{\max}$ or $-e_{\min}$ must imply that at least one of the numbers cannot be represented any more.
Hint: p does also count the leading digit.
Tip: Write down the normalized binary representation of the values, if it is not obvious for you.

/3P

Werte / *Values*: 2.25, $\frac{1}{8}$, 0.5, 16.5, 2^3

$F^*(\beta, p, e_{\min}, e_{\max})$ mit / *with*
$\beta = 2$, $p = 6$, $e_{\min} = -3$, $e_{\max} = 4$.

(b) Markieren die Werte, die eine exakte Darstellung in einem beliebigen (endlichen) normalisierten binären Fliesskommazahlensystem besitzen.

Mark the values that have an exact representation in an arbitrary (finite) normalized binary floating point number system.

/3P

◯ 123.4 ◯ 0.025 ◯ $1/10$ √ 1000.5 √ $1/16$ √ $1.5/32$

(c) Die Zahl in der linken Spalte der nachfolgenden Tabelle ist jeweils als Literal der Sprache C++zu verstehen. Berechnen Sie, was jeweils verlangt ist.

The number displayed to the left in the following table shall be considered a number literal in C++language. Compute what is requested.

/2P

0x7FF	Dezimalzahl / *decimal number*	=	2047
-16	Binärzahl mit 6 Bits im Zweierkomplement / *Binary number with 6 bits in two's complement*	=	110000

Aufgabe 4: EBNF I (6P)

Die folgende EBNF definiert erlaubte Anweisungen einer vereinfachten Programmiersprache.
Beantworten Sie die Fragen auf der rechten Seite.
Anmerkung: Leerschläge sind im Rahmen der EBNF bedeutungslos.

The following EBNF defines the allowed statements of a programming language. Answer the questions on the right hand side.

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

```
Statement      = Expression ';'.
Expression     = Simple [':' Simple].
Simple         = Designator | Integer.
Designator     = Identifier {'.' Identifier | '(' [ExpressionList] ')'}.
ExpressionList = Expression { ',' Expression }.

Integer        = Digit {Digit}.
Digit          = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' .

Identifier     = Letter {Letter}.
Letter         = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' .
```

(a) Folgende Zeichenkette entspricht einer gültigen Anweisung (statement) gemäss der EBNF:

The following string corresponds to a valid statement according to the EBNF:

/1P

```
a(2:5);
```

√ wahr / *true* ◯ falsch / *false*

(b) Folgende Zeichenkette entspricht einer gültigen Anweisung (statement) gemäss der EBNF:

The following string corresponds to a valid statement according to the EBNF:

/1P

```
a(b).c:d;
```

√ wahr / *true* ◯ falsch / *false*

(c) Folgende Zeichenkette entspricht einer gültigen Anweisung (statement) gemäss der EBNF:

The following string corresponds to a valid statement according to the EBNF:

/1P

```
a(3).3;
```

◯ wahr / *true* √ falsch / *false*

(d) Folgende Zeichenkette entspricht einer gültigen Anweisung (statement) gemäss der EBNF:

The following string corresponds to a valid statement according to the EBNF:

/1P

```
a(3):3;
```

√ wahr / *true* ◯ falsch / *false*

(e) Ändern Sie genau eine Produktionsregel der EBNF ab, so dass folgende Anweisung (statement) gültig wird.

Modify one and only one production rule of the EBNF such that the following statement becomes valid.

/2P

```
a3(c3):a4(c4);
```

Geänderte Zeile / *Modified line*:

```
Identifier = Letter {Letter | Digit}.
```

Aufgabe 5: EBNF II (8P)

Auf der nächsten Seite ist Code abgebildet, mit welchem identifiziert werden soll, ob eine Zeichenkette eine im Sinne obiger EBNF gültige Anweisung (Statement) darstellt. Folgender Code, welcher nur als Funktionsdeklaration vorliegt, soll als korrekt implementiert vorausgesetzt werden.

Consider the code on the next page that shall be used to check if a string provides a valid statement corresponding to the EBNF above. The following code that is only provided as a function declaration can be considered correctly implemented.

```
// POST: when the next available non-whitespace character equals c,
//     it is consumed and the function returns true,
//     otherwise the function returns false.
bool has(std::istream& is, char c);

// Integer = Digit {Digit}.
bool Integer (std::istream& is);

// Identifier = Letter {Letter}.
bool Identifier (std::istream& is);
```

/2P (a) Die Funktion `Expression` wird im Code anscheinend zweimal deklariert. Erklären Sie warum.

The function `Expression` seems to be declared twice in the code. Explain why.

Es gibt eine zirkuläre Abhängigkeit zwischen den Funktionen. Daher muss mindestens eine Funktion vor ihrer Definition deklariert werden. Sonst wäre sie in mindestens einer anderen Funktion nicht sichtbar.

/2P (b) Ergänzen Sie die Funktion `Designator`, so dass Sie die entsprechende EBNF-Zeile korrekt implementiert.

Complement function `Designator` such that it implements the corresponding EBNF line correctly.

/2P (c) Ergänzen Sie die Funktion `ExpressionList`, so dass Sie die entsprechende EBNF-Zeile korrekt implementiert.

Complement function `ExpressionList` such that it implements the corresponding EBNF line correctly.

/2P (d) Ergänzen Sie die Funktion `Expression`, so dass Sie die entsprechende EBNF-Zeile korrekt implementiert.

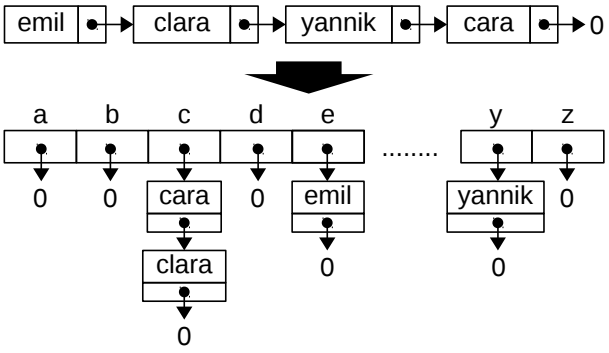
Complement function `Expression` such that it implements the corresponding EBNF line correctly.

```
bool Expression (std::istream& is);
// ExpressionList = Expression { ',' Expression }.
bool ExpressionList(std::istream& is){
    if (!Expression(is)) return false;
    while (has(is, ',')){
        if (!Expression(is)) return false;
    }
    return true;
}
// Designator = Identifier {'.' Identifier | '(' [ExpressionList] ')'}.
bool Designator(std::istream& is){
    if (!Identifier(is)) return false;
    while(true){
        if (has(is, '.')){
            if (!Identifier(is)) return false;
        } else if (has(is, '(')){
            bool ignore = ExpressionList(is);
            if (!has(is,')')) return false;
        } else
            return true;
    }
}
// Simple = Designator | Integer.
bool Simple(std::istream& is){
    return Integer(is) || Designator(is);
}
// Expression = Simple [':' Simple].
bool Expression(std::istream& is){
    if (!Simple(is)) return false;
    return !has(is,':') || Simple(is);
}
// Statement = Expression ';'.
bool Statement(std::istream& is){
    return Expression(is) && has(is, ';');
}
```

Aufgabe 6: Partial Bucketsort (8P)

Betrachten Sie das untenstehende Programm zum Teilsortieren einer Liste von Namen anhand ihres Anfangsbuchstabens. Die Namen bestehen ausschliesslich aus Kleinbuchstaben (`'a'-'z'`). Dabei werden die Elemente der Eingabe in Buckets (Gefässe) die ihrem Anfangsbuchstaben entsprechend einsortiert. Ein Array von verketteten Listen bildet die Buckets, wobei a=0, b=1, c=2, ... z=25. Die Eingabe erfolgt in der Form einer verketteten Liste L. Über jedes Element der Liste L wird von vorne her iteriert, dieses Element wird aus der Liste L entfernt und in die Liste des Zielbuckets zuvorderst einfügt.

Look at the program below. The program partially sorts a list of names according to the first letter of the name. The names consist entirely of lowercase characters (`'a'-'z'`). Each input element is inserted into a bucket that corresponds to its first letter. The bucket list is made up of an array of linked lists, where a=0, b=1, c=2, ... z=25. The input is a linked list. Implement the algorithm as follows: Iterate over each element of the input, remove it from the input list, and insert it as first element in the list of the correct bucket.



/2P (a) Vervollständigen Sie die Funktion `add_first` entsprechend den angegebenen Vor- und Nachbedingung.

Complete the function `add_first` *according to its pre- and postcondition.*

/3P (b) Vervollständigen Sie die Funktion `partial_bucketsort` mittels den Funktionen `remove_first` und `add_first`.

Complete the function `partial_bucketsort` *using the functions* `remove_first` *and* `add_first`.

/3P (c) Geben Sie die Ausgabe des Programms an (betrachten Sie die Funktionen `main` und `print`).

Give the output of the program (consider the functions `main` *and* `print`*).*

```
a: albert h: hubert l: lena lara
```

```
#include<iostream>
#include<string>

struct Node {
    std::string name;
    Node* next;
    Node(std::string _name, Node* _next) : name(_name), next(_next) {}
};
```

```
struct List {
    Node* first;
    List() : first(0) {}
    // PRE: non-empty list
    // POST: Returns and removes first element of list
    Node* remove_first() {
        Node* node = first;
        first = node->next;
        return node;
    }
    // PRE: valid list
    // POST: Adds node as first element of the list
    void add_first(Node* node) {
        node->next = first;
        first = node;
    }
};
void partial_bucketsort(List buckets[], List& unsorted) {
    while(unsorted.first != 0) {
        Node* node = unsorted.remove_first();
        int bucket = node->name[0]-'a';
        buckets[bucket].add_first(node);
    }
}
void print(List buckets[]) {
    for (List* list = buckets; list != buckets + 26; ++list) {
        if (list->first != 0) {
            std::cout << list->first->name[0] << ": ";
            for(Node* node = list->first; node != 0; node = node->next)
                std::cout << node->name << " ";
        }
    }
}
int main() {
    List buckets[26]; // 'a' - 'z'
    List unsorted;
    unsorted.first = new Node("lara", new Node("lena",
                     new Node("albert", new Node("hubert", 0))));
    partial_bucketsort(buckets, unsorted);
    print(buckets);
    return 0;
}
```

Aufgabe 7: Referenztypen und Pointer (7P)

Das Programm auf der rechten Seite definiert eine Datenstruktur für Datumsangaben und stellt weitere Operationen auf dieser Struktur bereit. Beantworten Sie für dieses Programm untenstehende Fragen.

The program on the right hand side defines a data structure to store dates and provides additional operations that operate on this structure. Answer the questions below with regard to this program.

/4P (a) In der untenstehenden Tabelle sehen Sie zwei verschiedene Ausgaben des Programms, sowie Spalten mit Markierungen, die jenjenigen in der Klasse `DateList` entsprechen. Füllen Sie die Tabelle mit Typdeklarationen oder Ausdrücken, so dass das Programm die entsprechende Ausgabe erzeugt (pro Zeile).

In the table below, you see two different outputs created by the program and columns with markers that point to locations within the class `DateList`. Fill in the markers with type declarations or expressions, such that the program creates the output as specified (per row).

Ausgabe: / *Output:*	A	B	C	D
1.4.2001 31.7.2013	`Date`	`Date& or Date`	`date`	`*it`
23.8.2017 31.7.2013	`Date*`	`Date&`	`&date`	`**it`

/2P (b) Betrachten Sie die Funktion `createDate`, welche eine Datumstruktur deklariert, initialisiert und zurückgibt. Beantworten Sie folgende Frage (je ein Satz): (i) Welchen Fehler hat diese Funktion und (ii) wie könnte man die Funktion korrigieren?

Look at the function `createDate` that declares and initializes a date structure und returns it. Answer the following questions (each one sentence): (i) What error does this function have, and (ii) how could the function be fixed ?

(i)
```
Returns pointer to deallocated object instance.
```

(ii)
```
Date* d = new(day, month, year);
```

/1P (c) Betrachten Sie die Funktionen `f1`, `f2`, `f3`, welche jeweils eine Datumsstruktur als Parameter übernehmen. Der Parameter ist ein Referenztyp oder ein Pointer mit unterschiedlicher const-Eigenschaft. Markieren Sie alle Funktionen die korrekt kompiliern.

Look at the function `f1`, `f2`, `f3` that receive a date structure as parameter. The parameters are either reference types or pointers and have different const properties. Mark those functions that compile correctly.

◯ `f1` ◯ `f2` √ `f3`

```
#include <iostream>
#include <vector>

struct Date {
  int day; int month; int year;

  void set(int d, int m, int y) { day = d; month = m; year = y; }
  void print() {
    std::cout << day << "." << month << "." << year << "\n";
  }
};

class DateList {
  std::vector< A > dates;
public:
  void add( B date) { dates.push_back( C ); }
  void print() {
    for(std::vector< A >::iterator it=dates.begin(); it!=dates.end(); ++it) {
      D .print();
    }
  }
};

int main() {
  DateList list; Date d1; Date d2;

  d1.set(1, 4, 2001);
  list.add(d1);
  d2.set(31,7, 2013);
  list.add(d2);
  d1.set(23, 8, 2017);

  list.print();
  return 0;
}

Date* createDate(int day, int month, int year) {
    Date d;
    d.set(day, month, year);
    return &d;
}

void f1(const Date& date) { date.year = 0; }

void f2(const Date* date) { date->year = 0; }

void f3(Date* const date) { date->year = 0; }
```