Informatik I D-ITET Prüfung 6.2.2017 Lösung *F. Friedrich*

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.*

Aufgabe	1	2	3	4	5	6	7	8	$\sum$
Punkte									
Maximum	7.5	7.5	10	8	7	10	10	10	70

1 Typen und Werte (Basistypen) *(7.5 Punkte)*

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

Die verwendeten Variablen sind wie folgt deklariert / initialisiert.

```
int n = 3;
int x = 0;
double d=30;
bool a=true;
bool b=false;
```

For each of the expressions on the right, provide its C++ type (0.5 P) and value (1 P)!

The used variables have been declared / initialized as shown above. If a value cannot be determined, write "undefined".

(a) `11 / 3 * n` — 1.5 P

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

(b) `--x + n--` — 1.5 P

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

(c) `0xf * 0xa` — 1.5 P

Typ/*Type*	Wert/*Value*
unsigned int or int	150

(d) `1 / 10 * d == d * 1 / 10` — 1.5 P

Typ/*Type*	Wert/*Value*
bool	false (or 0)

(e) `10.0 * (!a && !b) + 20 * (!a || !b)` — 1.5 P

Typ/*Type*	Wert/*Value*
double	20

1/2 Punkt für korrekten Typ, 1P. für korrekten Wert

2 Typen und Werte (Structs und Pointers) *(7.5 Punkte)*

Geben Sie für jeden der Ausdrücke auf der rechten Seite jeweils den C++-Typ (0.5 P) und Wert (1 P) an!

Die verwendeten Variablen haben Typ und Wert wie am Ende der Funktion `f`.

```
struct P {
     int a;
     unsigned int w;
     P* s;

     P(int A, unsigned int W, P* S): a(A), w(W), s(S)
        {}

     P(): a(4), w(13), s(0)
        {}
 };

void f(){
    P p;
    P* ps = new P(6, 20, &p);
    P* pt = new P(3, 17, ps);
    P& q = p;
    q.s = pt;

    int a = 10;
    int b = 2;
    int &c = a;
    ++c;

    int A[3] = {1,0,2};
    int *X = &A[0];

    // type and value of the variables at this point
}
```

For each of the expressions on the right, provide its C++ type (0.5 P) and value (1 P)!

Assume that the used variables have type and value as at the end of the function `f` above.

(a) `a` — 1.5 P

Typ/ *Type*	Wert/ *Value*
`int`	`11`

(b) `X[A[1]]` — 1.5 P

Typ/ *Type*	Wert/ *Value*
`int`	`1`

(c) `p.w` — 1.5 P

Typ/ *Type*	Wert/ *Value*
`unsigned int`	`13`

(d) `(*ps).a` — 1.5 P

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

(e) `p.s->a` — 1.5 P

Typ/ *Type*	Wert/ *Value*
`int`	`3`

1/2 Punkt für jeden korrekten Typ, 1P für jeden korrekten Wert

3 Programmausgaben *(10 Punkte)*

Betrachten Sie folgende Funktionen und geben Sie die fehlenden Nachbedingungen an. Die Nachbedingungen können informell sein, müssen aber die Ergebnisse und Effekte der Funktionen in Abhängigkeit von den Parametern vollständig charakterisieren.

Consider the following functions and provide the missing post conditions. The post conditions can be reasonably informal, but they must completely characterize the results and effects of the functions depending on the provided parameters.

(a) 2 P

```
// pre: n >= 0
// post: returns the sum of 0 ..  n ;
int s(int n){
    if (n>0)
        return n+s(n-1);
    return 0;
}
```

(b) 2 P

```
// pre: n > 0
// post: return number of digits of n ;
int d(int n){
    int res = 0;
    while (n > 0){
        n = n / 10;
        res++;
    }
    return res;
}
```

(c) 2 P

```
// pre: n > 1
// post: returns if n is a prime number ;
bool p(int n){
    int f=2;
    for (; n%f != 0; ++f);
    return n==f;
}
```

(d) 2 P

```
// pre: An array of characters provided by start pointer start
// and one-past-the-end pointer end
// post: reverse the string contained in the array from start ..
// end-1 ;
void r(char* start, char* end){
    char* it1 = start-1;
    char* it2 = end;
    while (++it1 < --it2){
        char t = *it1;
        *it1 = *it2;
        *it2 = t;
    }
}
```

(e) 2 P

```
// pre: positive integers a and b
// post: return greatest common divisor of a and b ;
int g(int a, int b){
    if (a == 0) return b;
    return g(b % a, a);
}
```

4 Fliesskommazahlen *(8 Punkte)*

Beantworten Sie die Fragen auf der rechten Seite. Diese Seite kann für Zwischenresultate verwendet werden.

Answer the questions on the right hand side. Use this page for intermediate results.

(a) Geben Sie pro Teilaufgabe jeweils die kleinstmögliche Präzisison des normalisierten Fliesskommasystems $F*(2, p, -3, 3)$ an, welche die **exakte** Representation der drei angegebenen Werte ermöglicht. 3 P

Please specify per sub task the smallest possible precision of the normalized floating point system $F*(2, p, -3, 3)$, which allows the **exact** *representation of the three numbers given.*

(i) $1.0011 \cdot 2^2$
$1.010101 \cdot 2^1$
$1.0111 \cdot 2^{-2}$

$p =$ 7

(ii) 9
$\frac{1}{2}$
$\frac{1}{8}$

$p =$ 4

(iii) $0.000111 \cdot 2^3$
$0.00001 \cdot 2^5$
$0.11 \cdot 2^{-2}$

$p =$ 3

(b) Binäre Fliesskommasysteme haben Lücken im Wertebereich. Gegeben sei $F*(2, p, e_{min}, e_{max})$. Geben Sie an, ob die Fliesskommazahlen A und B im Wertebereich von $F*$ direkt aufeinander folgen. Falls, ja, geben Sie die Grösse der Lücke D zwischen A und B als exakte Dezimalzahl an. 5 P

Binary floating point system have holes in their values range. Given the floating point system $F*(2, p, e_{min}, e_{max})$, specify whether the floating point numbers A and B follow each other directly in the value range of F. If yes, write the size of the hole D as exact decimal number.*

	p	e_{min}	e_{max}	A	B	aufeinanderfolgend *directly suceeding*	D
(i)	3	-3	3	$1.10 \cdot 2^2$	$1.11 \cdot 2^2$	yes	1
(ii)	4	-2	2	$0.001 \cdot 2^{-1}$	$0.001 \cdot 2^{-2}$	no	
(iii)	5	-3	3	$11.110 \cdot 2^{-1}$	$1.1111 \cdot 2^0$	yes	1/16 OR 0.0625

5 EBNF I *(7 Punkte)*

Die folgende EBNF definiert eine Sprache zur Beschreibung der Bedienung eines einfachen Verkaufsautomaten. Der Kunde kommt an ("arrive") zahlt Geld ("coin") fordert Schokolade ("choc") oder Getränk ("drink") an, oder möchte Geld zurück ("reject") und verlässt die Maschine wieder ("leave").

Beantworten Sie die Fragen auf der rechten Seite!

Anmerkung: Leerschläge sind im Rahmen der EBNF bedeutungslos.

The following EBNF defines a language for the description of the usage of a simple vending machine. The customor arrives ("arrive"), deposits money ("coin"), requests chocolate ("choc") or a drink ("drink") or reclaims the money ("reject"), and leaves the machine eventually ("leave").

Answer the questions on the right hand side.

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

```
S            = "arrive" Use.
Use          = "coin" SingleCredit | "leave".
SingleCredit = "choc" Use | "reject" Use | "coin" DoubleCredit.
DoubleCredit = "drink" Use | "reject" Use.
```

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

Folgende Zeichenkette entspricht einer gültigen Bedienung (S) der Maschine nach der EBNF:
The following string corresponds to a valid use (S) of the machine according to the EBNF:

```
arrive choc leave
```

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

Folgende Zeichenkette entspricht einer gültigen Bedienung (S) der Maschine nach der EBNF:
The following string corresponds to a valid use (S) of the machine according to the EBNF:

```
arrive coin choc leave
```

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

Folgende Zeichenkette entspricht einer gültigen Bedienung (S) der Maschine nach der EBNF:
The following string corresponds to a valid use (S) of the machine according to the EBNF:

```
arrive coin choc reject leave
```

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

Folgende Zeichenkette entspricht einer gültigen Bedienung (S) der Maschine nach der EBNF:
The following string corresponds to a valid use (S) of the machine according to the EBNF:

```
arrive coin coin coin reject leave
```

(e) Vervollständigen Sie folgende EBNF, so dass sie die selben Zeichenketten zulässt wie die EBNF auf der linken Seite. 3 P
Complement the following EBNF such that it permits the same strings as the EBNF on the left hand side.

```
S'      = "arrive" Use'.
Use'    = {"coin" SingleCredit'} "leave" OR "coin" SingleCredit Use |
          "leave".
SingleCredit' = "choc" | "reject" | "coin" DoubleCredit'.
DoubleCredit' = "drink" | "reject".
```

6 EBNF II *(10 Punkte)*

Betrachten Sie folgenden Code und beantworten Sie die Fragen auf der rechten Seite.
Consider the following code and answer the questions on the right hand side.

```
// POST: when the next available string at is equals s, it is consumed
//       and the function returns true, otherwise the function returns false.
bool has(std::istream& is, std::string s);
// S = "arrive" Use.
int S(std::istream& is){
    if (has (is, "arrive"))
        return Use(is);
}
// Use = "coin" SingleCredit | "leave".
int Use(std::istream& is){
    if (has(is, "coin"))
        return [1 + SingleCredit(is) [OR: SingleCredit(is)]] ;
    else if (has(is, "leave"))
        return [0];
    }
}
// SingleCredit = "choc" Use | "reject" Use | "coin" DoubleCredit.
int SingleCredit(std::istream& is) {
    if (has(is, "coin"))
        return [1 + DoubleCredit(is) [OR: DoubleCredit(is)]];
    else if (has(is, "reject"))
        return [-1 + Use(is) [OR: Use(is)]];
    else if (has(is, "choc"))
        return [Use(is) [OR: Use(is)+1]]);
}
// DoubleCredit = "drink" Use | "reject" Use.
int DoubleCredit(std::istream& is) {
    if (has(is, "reject"))
        return [-2 + Use(is) [OR: Use(is)]];
    else if (has(is, "drink"))
        return [Use(is) [OR: Use(is)+2]];
}
```

Die folgende main-Funktion soll zu einer Benutzung der Maschine die Anzahl Geldstücke ausgeben, welche die Maschine während der Benutzung behalten hat. Die Maschine behält eine Münze für jede verkaufte Schokolade und zwei Münzen für jedes verkaufte Getränk. Die Eingabe liegt gemäss der EBNF der vorigen Aufgabe am Eingabestrom vor und ist im Sinne der EBNF auch gültig.

Beispieleingabe (die unterstrichenen Münzen behält die Maschine):
`arrive coin choc coin reject coin coin drink coin reject leave`
Ausgabe: `Credit = 3`.

The following main function shall return the number of coins that the vending machine has kept during the use. The vending machine keeps one coin for each sold chocolate and two coins for each sold drink. The input is provided at the input stream according to the EBNF of the previous task and is valid according to the EBNF.

Example input (the underlined coins are kept by the machine):
`arrive coin choc coin reject coin coin drink coin reject leave`
Output: `Credit = 3`.

```
int main() {
    int credit = S(std::cin);
    std::cout << "Credit = " << credit << \n";
    return 0;
}
```

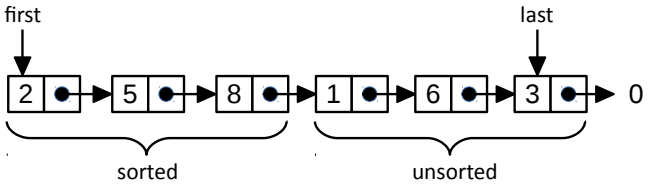
(a) Vervollständigen Sie den Code der Funktion `Use` entsprechend. 3 P
Complement the code of function `Use` accordingly.

(b) Vervollständigen Sie den Code der Funktion `SingleCredit` entsprechend. 4 P
Complement the code of function `SingleCredit` accordingly.

(c) Vervollständigen Sie den Code der Funktion `DoubleCredit` entsprechend. 3 P
Complement the code of function `DoubleCredit` accordingly.

7 Sortierung einer verketten Liste *(10 Punkte)*

`Struct Node` und `struct List` implementieren eine verkettete Liste. Vervollständigen Sie nebenstehende Funktionen, welche den folgenden Sortieralgorithmus implementieren: Die Liste wird unterteilt in einen sortierten und einen unsortierten Teil. Alle Listenelemente befinden sich initial im unsortierten Teil. Iteriere sequentiell über jedes Element der Liste. Für jedes Element e: 1. Finde das Element x mit dem kleinsten Wert im unsortierten Teil. 2. Vertausche die Werte von e und x mittels `swap`. 3. Element e gehört dann zum sortierten Teil der Liste.



`Struct Node` and `struct List` implement a linked list. Complete the functions on the right hand side that implement the following sorting algorithm: The list is subdivided into a sorted and an unsorted part. Initially, all list elements are in the unsorted part of the list. Iterate over each element of the list. For each element e of the list: 1. Find the element x with the smallest value, in the unsorted part of the list. 2. Swap the values of e and x using `swap`. 3. Element e now belongs to the sorted part of the list.

```
struct Node {
    int value;  Node* next;
    Node(int Value) : value(Value), next(0) { }
};
struct List {
    Node* first;
    Node* last;
    List() : first(0), last(0) { }
    void append(Node* node) {
        if (last == 0) {
            first = node;
        } else {
            last->next = node;
        }
        node->next = 0;
        last = node;
    }
    void swap(int &a, int &b) {
        int tmp = a; a = b; b = tmp;
    }
```

```
    // struct List (cont.)
    Node* find_smallest(Node* from);
    void  sort();
}
```

(a) Vervollständigen Sie die Funktion `List::find_smallest(Node* from)`, so dass sie das kleinste Element im Intervall (from,last) zurückgibt: 5 P

Complete the function `List::find_smallest(Node* from)`. It must return the smallest element in the interval (from,last):

```
// pre: from != null
Node* List::find_smallest(Node* from) {
    Node* smallest = from;
    Node* curr = from->next;
    while (curr != 0) {
        if ( curr->value < smallest->value ) {
            smallest = curr;
        }
        curr = curr->next;
    }
    return smallest;
}
```

(b) Vervollständigen Sie die Funktion `List::sort`, welche den Sortieralgorithmus implementiert: 5 P

Complete the function `List::sort` that implements the sorting algorithm:

```
void List::sort() {
    Node* curr = first;
    while (curr != 0) {
        Node* smallest = find_smallest(curr);
        swap(smallest->value, curr->value);
        curr = curr->next;
    }
}
```

8 Mengen-Datentyp *(10 Punkte)*

Die Klasse Set implementiert einen Datentyp zur Representation von Mengen. Elemente von 0 bis 9 können in einem Set gespeichert werden. Beantworten Sie die nachfolgende Fragen.

The class Set implements a datatyp to represent a set. The set can contain elements from 0 to 9. Answer the following questions.

```
class Set {
public:
    // POST: Empty set.
    Set() { for(int i = 0; i < 10; ++i) elems[i] = false; }
    // POST: Adds element to set.
    void add(int nr) { elems[nr] = true; }
    //POST: Removes element from set.
    void remove(int nr) { elems[nr] = false; }
    // POST: Returns the union of set this and set other.
    Set operator+(const Set& other) const;
    // POST: Returns true if set this and set are equal, otherwise false.
    bool operator==(const Set& other) const;
    // POST: Returns the intersection of set this and set other.
    Set operator*(const Set& other) const;

    bool elems[10];
}
```

(a) Implementieren Sie `Set::operator*`, welcher die Schnittmenge zweier Mengen zurückgibt. 4 P

Implement the `Set::operator*` that returns the intersection of two sets.

```
Set operator*(const Set& other) const {
    Set set ;
    for( int i = 0; i < 10; ++i ) {
        set.elems[i] = elems[i] && other.elems[i];
    }
    return set ;
}
```

(b) Wie kann man die Kapselung der Klasse Set verbessern, ohne Funktionalität einzubüssen? (Maximal 2 sehr kurze Sätze!) 2 P

How can the encapsulation of class Set be improved without loosing functionality? (Maximally 2 very short sentences!)

```
Make access to field elems private.  Provide getter and setter
functions.
```

(c) **Für diese Teilaufgabe dürfen Sie vom Set Datentyp ausschliesslich die Mitgliedsfunktionen verwenden!** 4 P

Geben Sie für jeden der folgenden Vergleichsoperatoren den entsprechenden Ausdruck an, so dass die angegebene Nachbedingungen der Operatoren jeweils erfüllt sind.

For this sub task, from data type Set you may only use its member functions!

For each of the following comparison operators specify the expression so that the given post-condition of each operator is fulfilled.

```
// POST: Returns true if set a and set b are unequal, otherwise false.
bool operator!=(const Set& a, const Set& b) const {
    return !(a == b);
}
// POST: Returns true if set a is a subset of set b.
bool operator <=(const Set& a, const Set& b) const {
    return a * b == a;
}
// POST: Returns true if set a is a superset of set b.
bool operator >=(const Set& a, const Set& b) const {
    return a + b == a;
}
// POST: Returns true if set a is a proper (strict)
//       subset of set b.
bool operator <(const Set& a, const Set& b) const {
    return a * b == a && !(a == b);
}
```