Informatik I D-ITET Self-Assessment IV, 11./12.12.2017 Lösung

Name, Vorname: . . . . .
Legi-Nummer: . . . . .

Diese Selbsteinschätzung dient Ihrer und unserer Orientierung. Sie wird eingesammelt, korrigiert und vertraulich behandelt. Sie hat keinen Einfluss auf eine spätere Leistungsbewertung. **Sie haben 15 Minuten Zeit.**

Das folgende Kleingedruckte finden Sie auch auf einer "scharfen" Prüfung.

Allgemeine Richtlinien:

1. Dauer der Prüfung: 15 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:

1. *Exam duration: 15 minutes.*
2. *Permitted examination aids: dictionary (for spoken languages). No sheets of your own! We will give you extra sheets on demand.*
3. *Use a pen (black or blue), not a pencil. Please write legibly. We will only consider solutions that we can read.*
4. *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.*
5. *There are no negative points for false answers.*
6. *If you feel disturbed by anyone or anything, let the supervisor of the exam know immediately.*
7. *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. *If you need to go to the toilet, raise your hand and wait for a supervisor.*
9. *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	$\sum$
Punkte				
Maximum	5	6	7	18

1 EBNF I *(5 Punkte)*

Die folgende EBNF definiert eine Sprache zur Beschreibung von kantengewichteten Bäumen. Ein geklammerter Ausdruck steht für einen Baum, ganze Zahlen bezeichnen Kantengewichte. Ein illustrierendes Beispiel ist unter der EBNF zu finden.

Beantworten Sie die Fragen auf der rechten Seite!

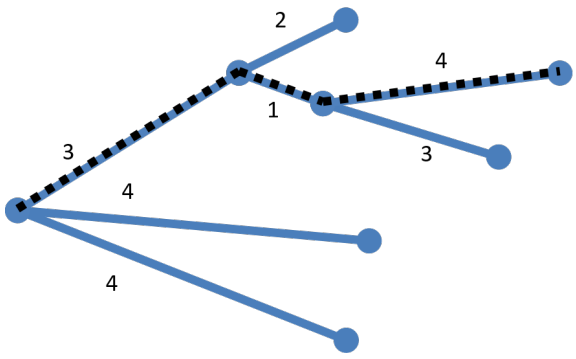
The EBNF displayed below defines a language for trees with weighted edges. An expression in parentheses stands for a tree, integer numbers designate edge weights. An illustrating example is displayed below the EBNF.

Answer the questions on the right side!

```
Tree = '(' Branches ')'.
Branches = Branch { Branch }.
Branch = Length | Length Tree.
Length = unsigned int.
```

Beispiel / *Example*:

(3 (2 1 (4 3)) 4 4) ⇔



Die gestrichelten Linien kennzeichnen Pfad mit maximaler Länge. Das ist nur für Aufgabe 2 relevant.

The dashed lines designate the path with maximal length. This is only relevant for problem 2.

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

Die folgende Zeichenkette ist ein gültiger Baum (`Tree`) nach der EBNF:
The following string is a valid tree (`Tree`) according to the EBNF:
`(10 20 30)`

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

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

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

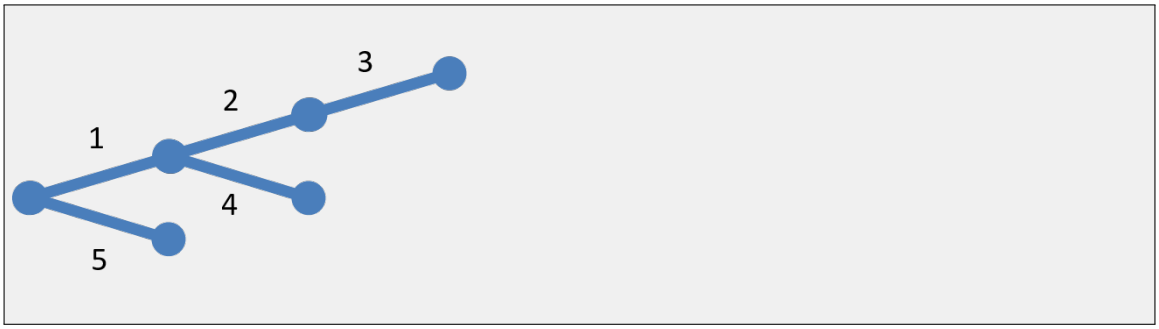
Die folgende Zeichenkette ist ein gültiger Baum (`Tree`) nach der EBNF:
The following string is a valid tree (`Tree`) according to the EBNF:
`(10 5 (20 30 40) 30 40 (5 2 1) 45)`

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

Die folgende Zeichenkette ist ein gültiger Baum (`Tree`) nach der BNF:
The following string is a valid tree (`Tree`) according to the BNF:
`10 (10 20 30)`

(e) Zeichnen Sie den Baum, welcher zu folgender Eingabe gehört. 1 P
Draw the tree that belongs to the following input.

(1 (2 (3) 4) 5)



2 EBNF II *(6 Punkte)*

Die folgende main-Funktion soll zu einem Baum, welcher gemäss der BNF der vorigen Aufgabe am Eingabestrom vorliegt, die maximale Pfadlänge ausgeben. Die Länge eines Pfades ist die Summe der Gewichte entlang des Pfades. Ein Pfad eines Baumes geht jeweils von der Wurzel des Baumes bis zu einem Blatt. Im Beispiel auf Seite 2 ist der Pfad mit maximaler Länge 3+1+4=8 gestrichelt eingezeichnet.

Vervollständigen Sie den Code der Funktionen `Tree`, `Branches` und `Branch` entsprechend (jeweils 2P).

The following main function shall return the maximal path length of a tree that is provided at the input stream according to the BNF defined in the previous task. The length of a path is given by the sum of its weights. A path of a tree starts from the tree root and ends at a leaf. In the example on page 2 the path with maximal length 3+1+4=8 is dashed.

Complement the code of functions `Tree`, `Branches` and `Branch` accordingly (2P each).

```
#include<iostream>

// code on the right hand side ---->

int main() {
    int length = Tree(std::cin);
    std::cout << "Maximal path length= " << length << \n";
    return 0;
}
```

```
// 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);

// Tree = '(' Branches ')'.
// Returns length of longest outgoing path.
int Tree (std::istream& is){
    char c;
    is >> c; assert(c=='(');
    length = Branches(is)                    ;
    is >> c; assert(c==')');
    return length;
}
// Branches = Branch { Branch }.
int Branches (std::istream& is){
    int length = Branch(is);
    while (lookahead(is) != ')'){
        int bLength = Branch(is);
        if (bLength > length) length = bLength;
    }
    return length;
}
// Branch = Length | Length Tree.
int Branch(std::istream& is){
    int length = Length(is);
    if (lookahead(is) == '(') return Tree(is)+length                    ;
    return length;
}
// Length = unsigned int.
// Returns integer present at input stream
int Length (std::istream& is);
```

3 Structs und Operatoren *(7 Punkte)*

Im Folgenden finden Sie ein Programm zum Rechnen mit 2-dimensionalen Vektoren. Ergänzen Sie die Definitionen der beiden Operatoren so, dass sich ein korrektes Programm ergibt! Das Skalarprodukt zweier Vektoren (a_x, a_y) und (b_x, b_y) ist die Zahl $a_x \cdot b_x + a_y \cdot b_y$; die Skalierung von (x, y) mit λ liefert den Vektor $(\lambda x, \lambda y)$.

In the following, you find a program for computing with 2-dimensional vectors. Complete the definitions of the two operators such that you get a correct program! The scalar product of two vectors (a_x, a_y) and (b_x, b_y) is the number $a_x \cdot b_x + a_y \cdot b_y$; scaling (x, y) by λ yields the vector $(\lambda x, \lambda y)$.

```
#include<iostream>

struct vector_2 { double x; double y; };

// POST: the scalar product of v and w is returned
double operator* (vector_2 v, vector_2 w)
{
    return v.x * w.x + v.y * w.y;
}
// POST: the scaled vector lambda * v is returned
vector_2 operator* (double lambda, vector_2 v))
{

   vector_2 result;
   result.x = lambda * v.x;
   result.y = lambda * v.y;
   return result;

}

int main()
{
  vector_2 v = {1.0, 2.0}; vector_2 w = {3.0, 4.0};
  std::cout << v * w << std::endl; // 11
  vector_2 u = 5.0 * v;
  std::cout << '(' << u.x << ", " << u.y << ')' << "\n"; // (5, 10)
  return 0;
}
```

Grading Scheme

Task 1 (EBNF)

Each subtask gets one point when completely correct, in particular this is true for (e). Trees drawn in a different orientation are, of course, also ok.

Task 2 (EBNF II)

Each box gets 2 points: [Tree] Branches(is) yields 2 points. Branches() with wrong parameters gets one point. Not calling Branches results in no point [Branches] Finding the maximum gets 2 points. This can be via if-clause or using std::max. A too early return results in no point. Adding instead of finding the max results in no point. Exception: when added the points here but taking max in [Branch] qualifies for one point. [Branch] Calling Tree() yields one point. Adding length to tree yields another point. Only length yields one point.

Task 3(Structs and Operators)

Scalar product yields 3 points. Scaled vector yields 4 points.

Inverted correction:

- baseline 7 points
- no operator overloading used -> -2 points
- each significant* syntactical mistake: -1 point
- no scalar product computed: -2p
- any wrong computation of the scalar product: -1p
- no scaling applied: -2p
- each error in scaling: -1p
- each forgotten return -1p
- type errors, forgotten references to the struct `vector_2`: -1p

Slightly wrong references to the names such as forgetting an underscore (`vector2`: instead of `vector_2`) are not significant.

Wrong types are significant. Wrong handling of element access is significant. A forgotten semicolon is not significant. Providing no semicolon at all, however, is significant.