Informatik D-MATH/D-PHYS Self-Assessment IV, 13.12.2016

Name, Vorname: .....

Legi-Nummer: ..... Übungsgruppe: .....

Diese Selbsteinschätzung dient Ihrer und unserer Orientierung. Sie wird eingesammelt, korrigiert und vertraulich behandelt. Sie hat keinen Einfluss auf eine spätere Leistungsbewertung. **Sie haben 20 Minuten Zeit.**

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

Allgemeine Richtlinien:

1. Dauer der Prüfung: 20 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: 20 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	4	$\sum$
Punkte					
Maximum	5	7	5	11	28

1 EBNF *(5 Punkte)*

Die EBNF unten definiert eine Sprache zur Beschreibung verschachtelter Listen, wie sie in Inhaltsverzeichnissen auftreten.[1] In der EBNF bezeichnet `text` eine beliebige (möglicherweise leere) Zeichenfolge ohne die Zeichen '<', '>' und ','. Ein solcher Text kann Leerzeichen enthalten. Beantworten Sie die Fragen unten!

```
list = "<" items ">"
items = { item ","} item
item = text [ list ]
```

The EBNF above defines a language for the description of nested lists as they occur in tables of contents.[2] *In the EBNF, `text` is an arbitrary (possibly empty) sequence of characters not containing '<', '>', and ',' . Such a text may contain spaces. Answer the questions below!*

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

Diese Liste ist gültig nach der EBNF: *This list is valid according to the EBNF:*

`<Languages<French,English,Swiss German>>`

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

Diese Liste ist gültig nach der EBNF: *This list is valid according to the EBNF:*

`<Food<Health Junk>>`

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

Diese Liste ist gültig nach der EBNF: *This list is valid according to the EBNF:*

`<Darth<Vader><Maul>>`

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

Diese Liste ist gültig nach der EBNF: *This list is valid according to the EBNF:*

`Table of Contents<Section 1,Section 2<Subsection 2.1>>`

(e) Gib eine kürzeste Liste an, die nach der EBNF gültig ist. *Provide a shortest list that is valid according to the EBNF.* 1 P

[1] {···} steht für eine endliche Anzahl (möglicherweise null) von Wiederholungen von ···, während [···] bedeutet, dass ··· optional ist.

[2] {···} stands for a finite number (possibly zero) of repetitions of ···, while [···] means that ··· is optional.

2 Structs und Operatoren *(7 Punkte)*

Unten finden Sie ein Programm zum Rechnen mit 2-dimensionalen Vektoren. Ergänzen Sie die Definitionen der beiden überladenen Operatoren so, dass sich ein korrektes Programm ergibt! Das Skalarprodukt zweier Vektoren (v_x, v_y) und (w_x, w_y) ist die Zahl $v_x \cdot w_x + v_y \cdot w_y$; die Skalierung von (v_x, v_y) mit λ liefert den Vektor $(\lambda \cdot v_x, \lambda \cdot v_y)$.

Below, you find a program for computing with 2-dimensional vectors. Complete the definitions of the two overloaded operators such that you get a correct program! The scalar product of two vectors (v_x, v_y) and (w_x, w_y) is the number $v_x \cdot w_x + v_y \cdot w_y$; scaling (v_x, v_y) with λ yields the vector $(\lambda \cdot v_x, \lambda \cdot v_y)$.

```
#include<iostream>

struct vector_2 { double x; double y; };

// POST: the scalar product of v and w is returned
                                                    
{
    ;
}
// POST: the scaled vector lambda * v is returned
                                                    
{
  
}
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;
}
```

---

3 Structs, Konstruktoren und New *(5 Punkte)*

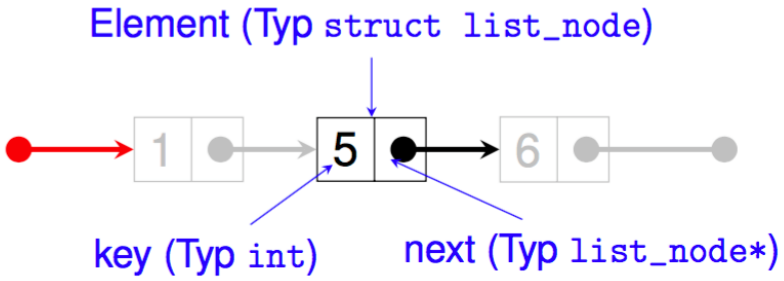
Unten finden Sie ein komplettes Programm. Ihre Aufgabe ist es, die Ausgaben der fünf angegebenen Zeilen zu bestimmen. Tragen Sie die Antworten direkt in die vorgegebenen Lücken ein! *Below you find a complete program. Your task is to determine the outputs of the five indicated lines. Fill your answers directly into the provided gaps!*

```
#include<iostream>

struct S {
  int v;
  S* n;

  S ()
    : v(5), n(0)
  {}

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

int main()
{
  S s;
  S* ps = new S (1, &s);
  S* pt = new S (9, new S (6, ps));

  std::cout << s.v;       // outputs
  std::cout << ps->v;     // outputs
  std::cout << pt->v;     // outputs
  std::cout << ps->n->v; // outputs
  std::cout << pt->n->v; // outputs

  delete ps; delete pt->n; delete pt;
  return 0;
}
```

4 Dynamische Datentypen *(11 Punkte)*

Auf der nächsten Seite finden Sie das Skelett einer Klasse `list`, die eine Folge ganzer Zahlen verwaltet. Jedes Element ist durch ein Objekt vom Typ `list_node` repräsentiert, das den Wert (`key`) sowie einen Zeiger auf das nächste Element (`next`) speichert (0 im Fall des letzten Elements). Eine Liste ist durch einen Zeiger auf ihr erstes Element repräsentiert (0 im Fall einer leeren Liste). Ergänzen Sie die Lücken so, dass sich korrekte und speicherleckfreie Definitionen der zwei Memberfunktionen `last` und `remove_second` ergeben! Die Assertions sollen dabei genau die angegebenen Vorbedingungen prüfen.

```
struct list_node {
    int key;              // value of the node
    list_node* next;      // pointer to next node
    ..
};
```



On the next page you find a skeleton of a class `list` for maintaining a sequence of integers. Each element is represented by an object of type `list_node` that stores the value (`key`) and a pointer to the next element (`next`) which is 0 in case of the last element. A list is represented by a pointer to its first element (0 in case of an empty list). Fill the gaps such that you obtain correct and memory-leak-free definitions of the two member functions `last` and `remove_second`! The assertions shall exactly check the given preconditions.

```
struct list_node {
    int key;                // value of the node
    list_node* next;        // pointer to next node
    ..
};

class list {
private:
  list_node* first_node; // pointer to the first node
public:
   ...
  // PRE:  *this is not empty
  // POST: the key of the last element in *this is returned
  int last () const
  {
     assert (                                    );
     const list_node* p =                                    ;
     while (                                    )
        ;
     return                                    ;
  }

  // PRE:  *this contains at least two elements
  // POST: the second element is removed from *this
  void remove_second ()
  {
        assert (                                    );
        assert (                                    );
        list_node* p =                                    ;
        ;
  }
};
```