

Einführung in die Programmierung

Klausur — WiSe 2023/24 — 27. März 2024

Nebentermin, Prüfungsnr. 23201019

Gruppe: Haskell

Unbedingt ausfüllen

Nachname

Vorname

Matrikelnummer

Identifikator (Beliebiges Wort zur Identifikation im anonymen Notenaushang)

Unterschrift

Hiermit bestätige ich meine Prüfungsfähigkeit.

Grundregeln

- Die Bearbeitungszeit der Klausur beträgt **120 Minuten**.
- Sie schreiben diese Klausur **vorbehaltlich** der Erfüllung der **Zulassungsvoraussetzung**. Das heißt: Wir werden Ihre Zulassung vor Korrektur prüfen; die Tatsache, dass Sie die Klausur mitschreiben, bedeutet keine implizite Zulassung.
- Es sind **keine Unterlagen** und auch **keine anderen Hilfsmittel** erlaubt.
- Benutzen Sie nur dokumentenechten (blauen oder zur Not schwarzen) **Kugelschreiber**. Bleistiftlösungen werden nicht gewertet.
- Es zählt die Antwort, die sich im dafür vorgesehenen Kästchen befindet! Soll eine andere Antwort gewertet werden, so ist diese **eindeutig** zu kennzeichnen und in der vorgesehenen grauen Box darauf hinzuweisen.
- Jegliches Schummeln, und auch der Versuch desselben, führt zum Ausschluss von der Klausur und einer Bewertung mit **5,0**.

Wird vom Korrektor/Prüfer ausgefüllt

Aufgabe	1	2	3	4	5	6	7	8	9	Σ
Punkte (max)	10	10	8	10	10	10	10	10	6	84
Punkte (erreicht)										

Note:

Aufgabe 1: Kurzfragen

(10 Punkte)

(a) Sichere Konversion

(2 Punkte)

Wann ist eine Konversion *sicher*?

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (a).

(b) Funktionssignatur

(2 Punkte)

Was ist eine Funktionssignatur?

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (b).

(c) Funktion vs. Methode

(2 Punkte)

Was ist eine Methode? Was ist insbesondere der Unterschied zu einer Funktion?

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (c).

(d) Stream-Zustände

(2 Punkte)

Welche Stream-Zustände gibt es? Geben Sie für jeden kurz die Semantik an.

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (d).

(e) protected

(2 Punkte)

Was macht das Schlüsselwort **protected**?

A large, empty rectangular box with a thin black border, intended for the student to write their answer to question (e).

Aufgabe 2: Fehlertypen

(10 Punkte)

Betrachten Sie die folgenden C++-Programme. Gehen Sie davon aus, dass am Anfang jedes Programms noch die Zeile `#include "std_lib_inc.h"` steht. Ansonsten sind die Programme vollständig. Gehen Sie zudem von den typischen Speichergrößen der Typen aus. Jedes Programm verursacht entweder einen oder gar keinen Fehler. Kreuzen Sie entsprechend an.

Hinweis: Das Nicht-ankreuzen einer Teilaufgabe gibt 0 Punkte für diese Teilaufgabe; falsche Kreuze führen zu Punkteabzug. Die Gesamtaufgabe kann jedoch nicht weniger als 0 Punkte bringen.

	kein Fehler	Fehler zur...			Logikfehler
		Kompilierzeit	Linkzeit	Laufzeit	
<pre>int main() { int x = 0; if(x != 0) { return 1; } return 0; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>struct Mensch { void hallo() { cout << "Hallo!"; } }; struct InformatikerIn : public Mensch { }; int main() { InformatikerIn sarah; sarah.hallo(); // Gibt "Hallo!" aus }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>int main() { int x = 1; for(int i = 1; i <= 100; ++i) { x = x * 10; // berechne schrittweise 10^100 cout << x << " "; } }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>int main() { vector<int> v = {1, 2, 3, 4}; for(int i = 1; i < v.size() + 1; ++i) { cout << v.at(v.size() - i) << '\n' ; } // Ausgabe von v rueckwaerts }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<pre>template<typename T> T returnInput(T x); int main() { int x = 42; int y = returnInput(x); if(x == y) { return 0; } // y ist Kopie von x return 1; }</pre>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Aufgabe 3: Programmsemantik

(8 Punkte)

Geben Sie exakt die jeweilige Ausgabe der folgenden C++-Programme an. Fügen Sie keine Zeichen hinzu oder lassen sie weg. Sie können davon ausgehen, dass keine Compiler-Optimierung stattfindet.

(a) (2 Punkte)

```
#include "std_lib_inc.h"
int f(int a, int& b) {
    b *= 3;
    return b - a;
}
int main()
{
    int x = 2;
    int y = 3;
    y = f(1, x);
    f(3, y);
    x = f(2, x);
    cout << x << y;
}
```

Ausgabe:

(b) (2 Punkte)

```
#include "std_lib_inc.h"
int f(double x) {
    return x;
}
int main()
{
    char a = 7;
    double b = 8.2;
    int c = 9.3;
    double d = f(a);
    double e = f(b);
    double g = f(c);
    cout << b << c << d << e << g;
}
```

Ausgabe:

(c) (2 Punkte)

```
#include "std_lib_inc.h"
struct A {
    A() {cout << "0";}
    ~A() { cout << "Y"; }
    A(int) { cout << "1";}
    A(double, string) {
        cout << "2";
    }
};
int main()
{
    const A a {5};
    A b {};
    A c(3);
    const A& d = b;
    A e(2.7, "Hallo");
}
```

Ausgabe:

(d) (2 Punkte)

```
#include "std_lib_inc.h"
int f(int x) {
    if(x > 42)
        throw runtime_error("X");
    if(x < 3)
        throw runtime_error("Y");
    return -x;
}
int main() {
    try {
        cout << f(17);
        cout << f(50);
        cout << f(0);
    } catch(runtime_error& e) {
        cout << e.what();
    }
    cout << "Z";
}
```

Ausgabe:

Aufgabe 4: Scopes

(10 Punkte)

(a) Scopes finden

(6 Punkte)

Markieren Sie alle Scopes im folgenden Programmcode und benennen Sie die Scope-Art. Sie können das entweder durch Umkreisen/Umklamern & Beschriften tun oder als Liste mit Zeilennummern.

```
1  #include "std_lib_inc.h"
2
3  namespace A {
4
5  class B {
6      int b_;
7  public:
8      int b() { return b_; }
9  };
10
11 int y = 7;
12 }
13
14 int main() {
15     using A::y;
16     while(y > 10) {
17         A::B b;
18         cout << b.b();
19     }
20     return 0;
21 }
```

(b) Scopes nachvollziehen

(4 Punkte)

Geben Sie exakt die Ausgabe des folgenden C++-Programms an. Fügen Sie keine Zeichen hinzu oder lassen sie weg. Sie können davon ausgehen, dass keine Compiler-Optimierung stattfindet.

```
#include "std_lib_inc.h"

namespace A {
    int y = 1;
}

int main()
{
    int y = A::y;
    {
        A::y *= 2;
        int y = A::y + 2;
        for(int i = 0; i < 3; ++i)
        {
            cout << i * A::y + y;
        }
    }
    cout << A::y - y;
}
```

Ausgabe:

Aufgabe 5: Rekursion

(10 Punkte)

Betrachten Sie die folgende Funktion $f: \mathbb{Z} \rightarrow \mathbb{Z}$.

$$f(x) = \begin{cases} f(\lfloor x/2 \rfloor) + \lfloor f(x-7)/2 \rfloor & x > 0 \text{ und ungerade} \\ 3 \cdot f(\lfloor x/4 \rfloor) & x > 0 \text{ und gerade} \\ 1 & x \leq 0 \end{cases}$$

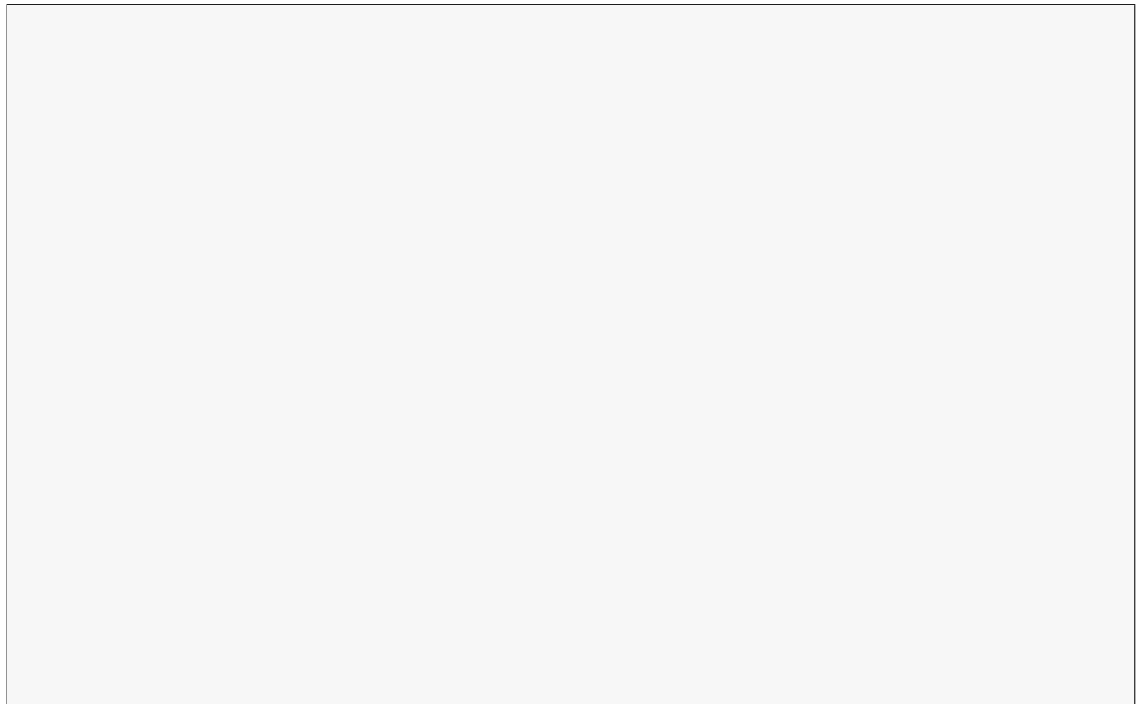
Hinweis: $\lfloor \dots \rfloor$ bezeichnet Abrunden.

(a) Rekursive Funktion

(6 Punkte)

Schreiben Sie eine rekursive C++-Funktion, die die obige Funktion berechnet. Sie dürfen hierfür keine Header-Dateien laden.

```
int f(int x) {
```

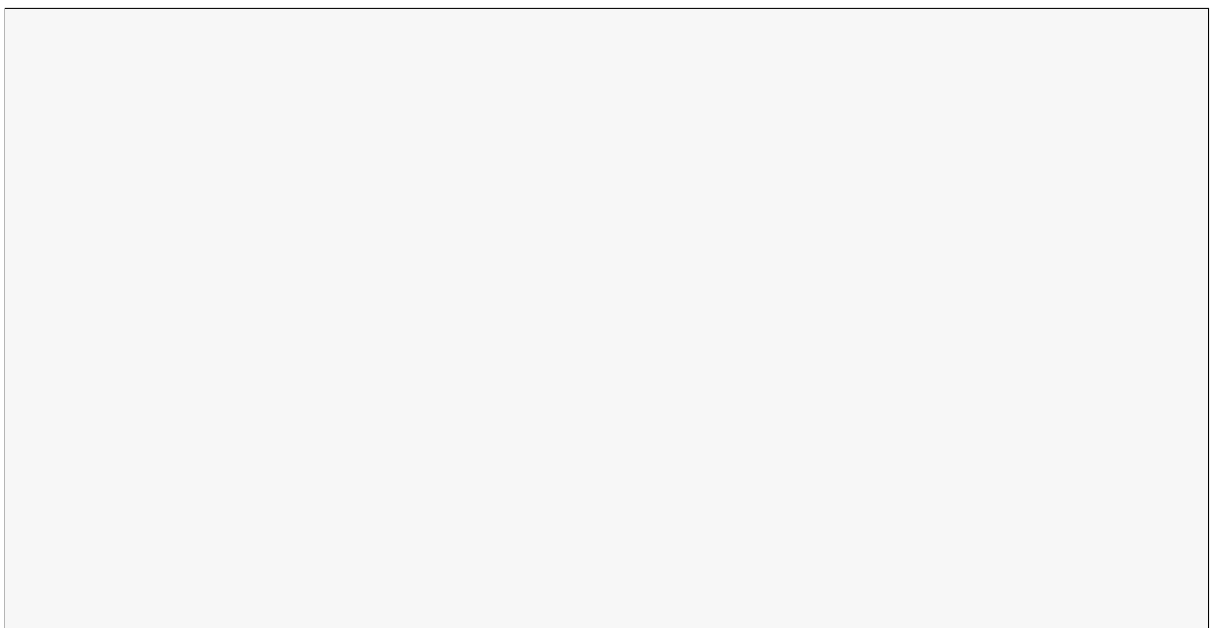


```
}
```

(b) Rekursive Berechnung

(4 Punkte)

Was ist $f(15)$ mit Rechenweg?



Aufgabe 6: Stream-Modell

(10 Punkte)

Schreiben Sie einen Eingabe-Operator für das folgende Struct Schlange:

```
#include "std_lib_inc.h"

enum class Muster {
    Gestreift, // Koerper besteht aus mehreren =
    Kariert,   // Koerper besteht aus mehreren #
    Gewellt   // Koerper besteht aus mehreren v
};

struct Schlange {
    string rufname; int laenge; Muster muster;
};

// Wandelt Eingabezeichen '=', '#' oder 'v' in Muster-Objekt
// Muster::Gestreift, Muster::Kariert oder Muster::Gewellt um.
Muster charToMuster(char);
```

Die Eingabe besteht aus dem (nicht-leeren) rufnamen, gefolgt von einem Leerzeichen und einer grafischen Repräsentation der Schlange. Diese grafische Repräsentation enthält laenge > 0 viele identische Zeichen, gewählt gemäß des musters der Schlange. Die Eingabe wird mit einem Zeilenumbruch beendet. Die Funktion charToMuster oben dürfen Sie verwenden.

Beispiel

Eingabe: „Hasso =====“ (ohne Anführungszeichen), gefolgt von Zeilenumbruch.
Schlange: rufname „Hasso“, laenge 5 und muster Gestreift.

```
#include "std_lib_inc.h"
```

Aufgabe 7: Listen

(10 Punkte)

Eine einfach verkettete Liste ist definiert durch die folgende Struct:

```
#include "std_lib_inc.h"

struct Eintrag {
    string wort;
    Eintrag* nachfolger;

    Eintrag(const string& w, Eintrag* n = nullptr)
        : wort(w), nachfolger(n) {}
};

Eintrag* finde(Eintrag* liste, string suchwort);
Eintrag* loesche(Eintrag* eintrag);
```

Implementieren Sie die Funktionen `finde` und `loesche`: Die Funktion `finde` erhält einen Zeiger auf den ersten Eintrag einer Liste und ein `string` `suchwort` und gibt einen Zeiger auf einen Eintrag `e` zurück, sodass `e.wort == suchwort` gilt. Ist das `suchwort` nicht enthalten, soll `nullptr` zurückgegeben werden. Die Funktion `loesche` löscht das Element auf das `eintrag->nachfolger` zeigt und gibt einen Zeiger auf `eintrag` zurück. Die Listeneigenschaften der restlichen Liste sind nach der Ausführung weiterhin sichergestellt. Ist `eintrag == nullptr`, so wird nur `nullptr` zurückgegeben.

Aufgabe 8: Templates

(10 Punkte)

Ein Modus (oder Modalwert) einer Menge ist ein Objekt, welches in dieser Menge am häufigsten vorkommt. Beispielsweise ist 3 ein Modus der Menge {1, 2, 3, 3, 3, 4, 4}. Beachten Sie, dass der Modus einer Menge im Allgemeinen nicht eindeutig ist.

Schreiben Sie ein Funktionstemplate `modus`, welches einen Modus der Objekte in einem `vector` zurückgibt. Das Template hat einen Templateparameter `T`, bekommt einen `const-Referenz-Parameter` vom Typ `vector<T>` übergeben und gibt ein Objekt vom Typ `T` zurück. Ist der übergebene `vector` leer, so soll eine `Exception` ausgegeben werden.

Hinweis: Sie dürfen für die Objekte in dem `vector` annehmen, dass ein `operator==` definiert ist.

Beispiel

Übergeben: `std::vector<int>` mit Inhalt (1, 4, 5, 2, 3, 1, 2, 2, 3, 4, 3)
(Eine mögliche) Rückgabe: 2

```
#include "std_lib_inc.h"
```

Aufgabe 9: Vererbung

(6 Punkte)

Betrachten Sie das folgende Teilprogramm:

```
#include "std_lib_inc.h"

class SpielerIn {
    string name_;
public:
    SpielerIn(string name) : name_(name) {}
    string name() { return name_;}
};

***

int main()
{
    BrettspielerIn spielerin1("Hanna", 22);
    cout << spielerin1.name() << " ist " << spielerin1.alter()
         << " Jahre alt und beginnt.\n";
}
```

Schreiben Sie eine Klasse `BrettspielerIn`, die von `SpielerIn` erbt und an der Stelle `***` oben eingefügt wird. Sie soll ein privates Attribut `alter_` enthalten und die öffentliche Getter-Methode `alter`. Beim Ausführen des Programms soll „Hanna ist 22 Jahre alt und beginnt.“ (gefolgt von einer neuen Zeile) ausgegeben werden.

Raum für Notizen:

Viel Erfolg!