Probeklausur im SS2016
Informationstechnik

Institut für Technik der Informationsverarbeitung – ITIV
Prof. Dr.-Ing. E. Sax

Informationstechnik
Datum: 21.07.2016
Name: Max Mustermann
Matr. Nr.: 20160721
ID: 1

Hörsaal: Neue Chemie
Platz: 1

Hinweise zur Klausur

Hilfsmittel:

- Erlaubte Hilfsmittel sind Lineal und eine Notizen-/Formelsammlung, handgeschrieben DIN A4 Blatt (zweiseitig beschrieben).
- Verwenden Sie zum Bearbeiten der Aufgaben nur dokumentenechte Schreibgeräte – keinen Bleistift, keine Rotstifte!
- Alle nicht genannten Hilfsmittel sind untersagt. Dies beinhaltet jegliche Kommunikation mit anderen Personen sowie die Benutzung eines Taschenrechners.

Klausurdauer:
Die Prüfungsdauer für die Klausur beträgt 60 Minuten.

Klausurunterlagen:
Die Klausurunterlagen bestehen aus insgesamt 20 Seiten Aufgabenblättern (inklusive dieses Titelblatt, 6 Aufgabenblöcke und 1 zusätzliches Lösungsblatt) und 1 DIN A4 Blatt (handschriftlich).
Bitte prüfen Sie vor der Bearbeitung der Aufgaben auf jeder Seite oben Ihre Matrikelnummer sowie Ihre ID und zusätzlich Ihren Namen auf der ersten Seite.
Falls Sie zusätzliche Blätter zur Lösung der Aufgaben benötigen, fragen Sie nach zusätzlichem Lösungspapier bei der Aufsicht. Vermeiden Sie generell das Beschreiben der Rückseiten. Die Verwendung von eigenen Blättern ist nicht erlaubt. Geben Sie zu jeder Aufgabe ein detailliertes Rechenweg an. Lösungen ohne Rechenweg können trotz richtigem Ergebnis zu Punktabzug führen.

Klausurabgabe:
In den letzten 30 Minuten der Klausur ist eine vorzeitige Abgabe der Klausur nicht möglich. Am Ende der Klausur bleiben Sie bitte sitzen. Alle Aufgaben- und Lösungsblätter sowie dieses Deckblatt sind in den ausgehändigten Umschläge abzugeben. Diese werden von der Aufsicht eingesammelt.

Aufgabe 1: Rechnerarchitekturen

Aufgabe 1.1: Allgemeines

A) Zeichnen Sie eine interne Architektur nach von Neumann. Beschriften Sie dabei die einzelnen Komponenten und beschreiben Sie jeweils deren Hauptfunktion.

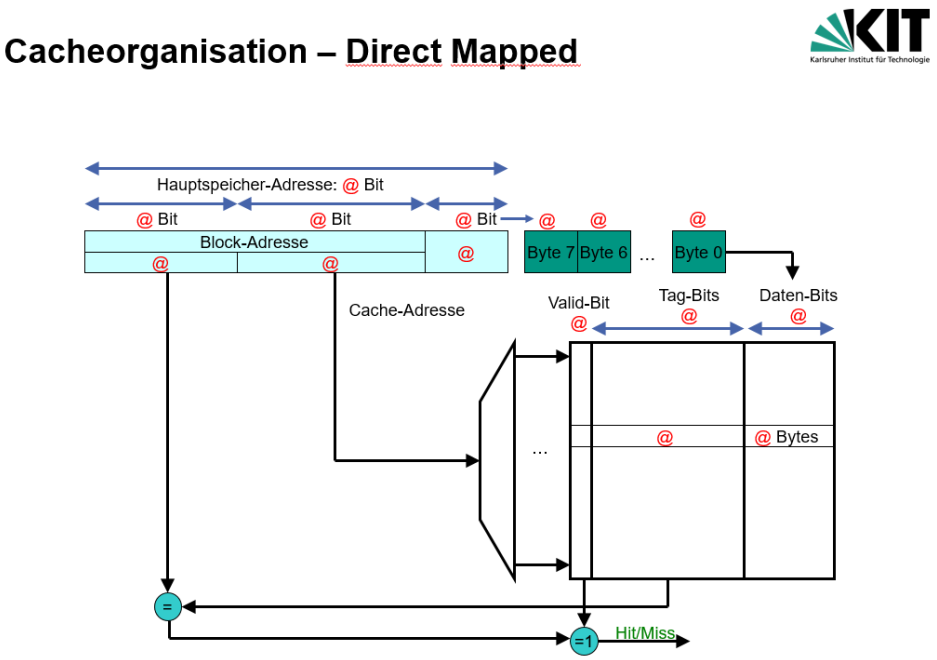
B) Nennen Sie die verschiedenen Speicherformen eines Rechners und sortieren Sie diese nach ihrer Zugriffszeit (schnell - langsam).

Aufgabe 1.2: Cacheorganisation

Angenommen, ein Prozessor hat einen Hauptspeicher der Größe 1 GByte, wobei jedem einzelnen Byte im Speicher eine 32 Bit lange Adresse zugeordnet wird. Zusätzlich verfügt der Prozessor über einen Cache Baustein in SRAM Technologie, der 512 KByte (nur Daten) vom Hauptspeicher aufnehmen kann (unabhängig vom notwendigen Speicher zur Speicherung der weiteren Bits, wie z.B. Tag). Der Cache hat eine Blockgröße (Zeilengröße) von 8 Byte. Zusätzlich wird auf dem Cache für jeden Block ein Valid-Bit vorgesehen.

A) Nennen Sie die drei Möglichkeiten der Cacheorganisation.

B) Berechnen Sie die Bitbreite von Offset, Index und Tag für einen Direct Mapped Cache und beschriften Sie die Grafik unten indem Sie alle @-Zeichen durch Namen bzw. Werte ersetzen.



Aufgabe 2: Verständnisfragen

A) Bringen Sie die folgenden Begriffe in eine logische Reihenfolge und stellen Sie die Beziehungen grafisch dar:
Linker - Header-Dateien - Standard-Bibliotheken - ausführbare Datei - Compiler - Quelldatei - Editor - Objektdatei - weitere Bibliotheken, Objektdateien.

B) Gegeben ist der folgende kurze Programmausschnitt. Nehmen Sie an, dass das Programm korrekt und fehlerfrei kompiliert worden ist. Welches sind die Ausgaben auf dem Bildschirm aufgrund der dargestellten Zeilen C++ Programmcode?

```
int main() {
     int i = 5;
     cout « 5 / 2 « endl;
     cout « 5.0 / 2 « endl;
     cout « i++ « endl;
     cout « ++i « endl;
}
```

C) Gegeben ist das folgende C++ Programm. Dieses enthält Programmzeilen, welche sowohl beim Kompilieren, als auch zur Laufzeit zu Fehlern führen. Benennen und begründen Sie die vier Fehler im C++ Programmcode.

```
#include <iostream>

int main() {
     int v = 3;
     array1[5] = {1, 2, 3, 4, 5};
     v = array1[5];
     cout « "Hallo" « Welt;
     return 0;
}
```

D) Welchen Wert speichert die Variable `x` nach Ausführung der folgenden Schleife ?

```
int x = 0;
for( int i = 0; i < 4; x += i++ );
```

E) Zeichnen Sie ein Nassi-Shneiderman Diagramm, welches einen Algorithmus beschreibt, um die Fakultät einer Zahl zu berechnen. Die Zahl soll dabei von der Tastatur eingelesen werden und vor der Berechnung überprüft werden, ob Sie gültig (größer oder gleich null) ist. Bei einer negativen Zahl soll eine Fehlermeldung, ansonsten soll das Ergebnis in einer Schleife berechnet und ausgegeben werden.

Aufgabe 3: Objektorientierung

A) Benennen und erklären Sie fünf Fehler im folgenden C++-Code:

```
class Base {
private:
     int value;

protected:
     void set_value(int x) const {
          value = x;
     }

public:
     int get_value() {
          return value;
     }
};

class Derived :  public Base {
public:
     Derived(int x) {
          set_value(x * 10);
     }

     int get_value() const {
          return vlaue * 10;
     }
};

int main() {
     Base b;
     b.set_value(100);

     Derived d;
     d.get_value();

     Derived d2 = b;

     return 0;
}
```

B) Erklären Sie die Bedeutung von `public` als Schlüsselwort in Bezug auf Zugriffsrechte in C++.

C) Erklären Sie die Bedeutung von `delete` als Schlüsselwort in C++.

D) Erklären Sie die Bedeutung von Konstruktoren und Destruktoren in Klassen in C++.

Aufgabe 3.1: Fahrzeuge

Gegeben ist folgender C++-Code:

```
#include <iostream>
#include <string>
using namespace std;

class Fahrzeug {
protected:
     int gesamt_masse, top_speed;
public:
     virtual ~Fahrzeug() {}

     int get_masse() const {
          return gesamt_masse;
     }

     int get_speed() const {
          return top_speed;
     }

     virtual string get_typ() const {
          return "Fahrzeug";
     }

     void print();
};

class PKW : public Fahrzeug {
public:
     PKW(int masse, int speed) {
          gesamt_masse = masse;
          top_speed = speed;
     }

     virtual get_typ() const {
          return "PKW";
     }
};

class LKW : public Fahrzeug {
public:
     LKW(int masse, int speed) {
          gesamt_masse = masse;
          top_speed = speed;
     }

     virtual string get_typ() const {
          return "LKW";
     }
};
```

Ein Aufruf der `print()` Methode gebe folgende Zeilen auf der Konsole aus:

```
Gesamtmasse:  1000 kg
Topspeed:  90 km/h
Typ:  PKW
```

A) Implementieren Sie die Methode `print()` so, dass abhängig vom zugrundeliegenden Objekt die Gesamtmasse, der Top-Speed und der Typ mitsamt der Einheiten auf der Konsole ausgegeben werden.

B) Gegeben sei nun folgende Ausgabe der `main()` Funktion auf der Konsole:

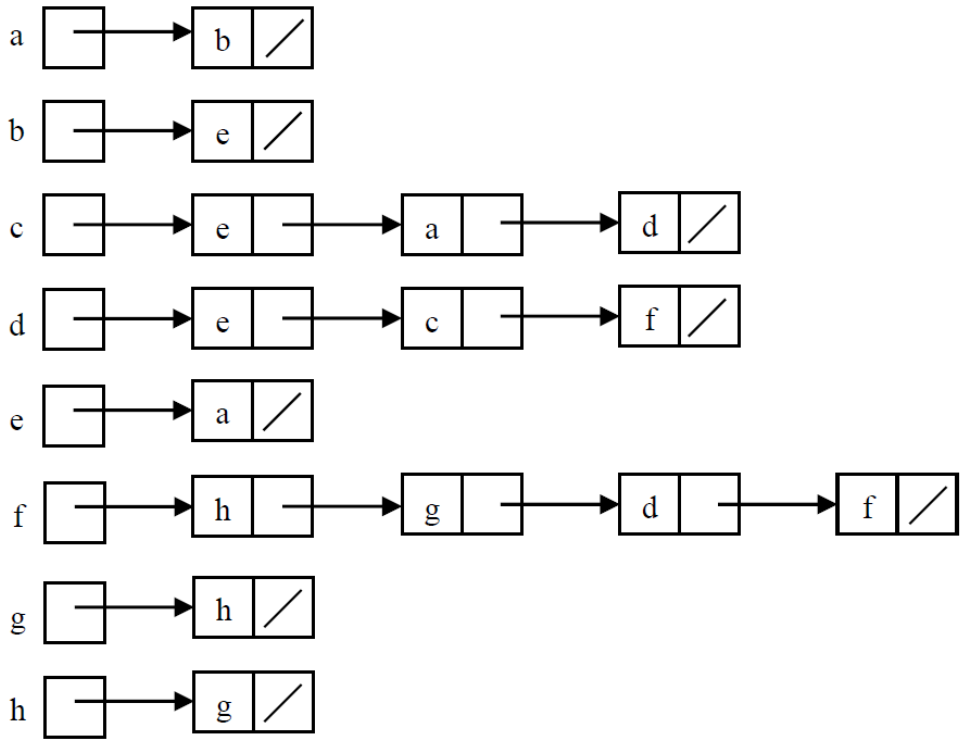
```
Fahrzeug
PKW
LKW
Gesamtmasse:  2000 kg
Topspeed:  250 km/h
Typ:  PKW
Gesamtmasse:  10000 kg
Topspeed:  100 km/h
Typ:  LKW
```

Ergänzen Sie die `main()` Funktion so, dass die obige Ausgabe auf der Konsole dargestellt wird. Vermeiden Sie dabei redundanten Code. Es dürfen ausschließlich Methodenaufrufe für die Konsolenausgaben verwendet werden!
Verwenden Sie die gegebenen Zeiger auf Fahrzeuge. Die Objekte sollen dabei dynamisch auf dem Heap erzeugt werden. Sorgen Sie außerdem dafür, dass der verwendete Heap-Speicher vor Beendigung der `main()` Funktion wieder freigegeben wird.

```
int main() {
     Fahrzeug* f;
     Fahrzeug* pkw;
     Fahrzeug* lkw;

     //Fügen Sie hier Ihren C++ Code ein



}
```

Aufgabe 4: Tiefensuchalgorithmus

Gegeben ist die folgende Adjazenzliste, welche den Graph Z beschreibt:



Gegeben ist außerdem der Pseudocode des Tiefensuchalgorithmus. Dabei ist `Z` der gegebene Graph und `Adj[u]` sind alle Nachfolger des Knotens `u`.

```
DFS(Z)
     for alle Knoten u in Z
     do farbe[u] = weiss
          vater[u] = NULL
     zeit = 0
     for alle Knoten u in Z
     do if farbe[u] == weiss
          then DFS-VISIT(u)

DFS-VISIT(u)
     farbe[u] = grau; zeit = zeit + 1
     startTime[u] = zeit
     for alle Knoten v aus Adj[u]
     do if farbe[v] == weiss
          then vater[v] = u
               DFS-VISIT(v)
     farbe[u] = schwarz; zeit = zeit + 1
     endTime[u] = zeit
```

A) Beschreibt die Adjazenzliste einen gerichteten oder einen ungerichteten Graphen? Begründen Sie Ihre Antwort.

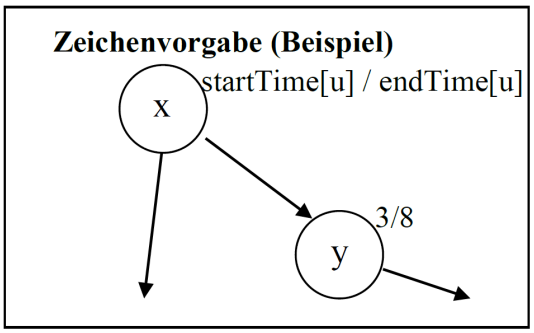
B) Wie viele Knoten und wie viele Kanten hat dieser Graph?

C) Bestimmen Sie den Grad des Knotens `f`.

D) Bestimmen Sie die Mächtigkeit der Menge der direkt benachbarten Knoten von `f`.

E) Geben Sie eine beliebige einfache Kantenprogression der Länge 4 an.

F) Wenden Sie den Tiefensuchalgorithmus (DFS) auf die Adjazenzliste an, um festzustellen, ob ausgehend vom Knoten `c` der Knoten `g` erreichbar ist.
Arbeiten Sie den DFS-Algorithmus vollständig (alle Konten mit Entdeckungs- und Endzeit) ab. Zeichnen Sie den entstehenden Tiefensuchbaum/-wald ausgehend von Startknoten `c`. Die Entdeckungsreihenfolge ist dabei bestimmt durch die Adjazenzliste.
Bestimmen Sie für jeden Knoten die Entdeckungszeit startTime[`u`] und die Endzeit endTime[`u`].



Aufgabe 5: Projektmanagement

A) Was versteht man unter Projektmanagement?

B) Ein Projekt gliedert sich im Allgemeinen in vier Phasen: Projektdefinition, Projektplanung, Projektdurchführung und Projektabschluss. Erklären Sie die Inhalte der vier Phasen.

C) In den Prozessen für den Entwurf von Software unterscheidet man die Elemente Software-Entwurf, Unit-Tests und Integrationstests. Warum wir hier zwischen Unit- und Integrationstest unterschieden. Nennen Sie einen entscheidenden Vorteil.

D) Ordnen Sie die pragmatische Vorgehensweise für die Erstellung von kleineren SW-Aufgabenstellungen dem Ablauf nach an. Beginnen Sie mit 1 für den ersten Schritt.

	Verschaffe dir Klarheit über die Aufgabenstellung
	Vervollständige die Dokumentation des entworfenen Programmsystems
	Finde den grundsätzlichen Aufbau der Eingabeobjekte bzw. des Eingabestroms
	Entwickle eine Lösungsidee für die jeweilige (Teil-)Aufgabe
	Transformiere die Lösungsidee in eine algorithmische Form
	Überprüfe die Korrektheit der algorithmischen Form der Lösung
	Fälle eine Entscheidung über die Entwurfsstrategie
	Transformiere den Algorithmus in ein Programm und überprüfe dessen syntaktische Korrektheit
	Teste das den Algorithmus repräsentierende Programm
	Beurteile die Qualität des Algorithmus und überarbeite ihn gegebenenfalls
	Definiere die (Algorithmen-)Schnittstelle(n)

Aufgabe 6: Hardwarenahe Programmierung

Aufgabe 6.1: Allgemeine Fragen

A) Was bezeichnet man als Register in der Informationstechnik?

B) Erklären Sie Interrupt und die Interrupt-Service-Routine (ISR).

Aufgabe 6.2: Register-Programmierung

Für ein Home-Entertainment-System sollen Sie eine Ansteuerung der verschiedenen Eingänge programmieren. Hierfür sollen über Taster die I/O-Pins der entsprechenden Treiber-Endstufen ausgewählt werden. Die Taster sind bereits mit den I/O-Pins an Port A, die Treiber-Endstufen mit den I/O-Pins an Port B verlötet. Im nachfolgenden ist ein Ausschnitt des Datenblatts und der zu realisierenden Funktionen dargestellt.

Konfiguration
Zur Konfiguration des Home-Entertainment-Systems müssen die Pins des Mikrocontrollers entsprechend konfiguriert werden. Hierfür müssen die Ports über entsprechende Register als Ein- bzw. Ausgang konfiguriert werden. Zur Konfiguration der Ports müssen die Werte aus Tabelle 6.1 in das jeweilige Mode-Register (MR) geschrieben werden. Durch die Konfiguration des Mode-Registers wird der jeweilige Port mit allen I/O-Pins als Ein- oder Ausgang konfiguriert.

Wert	Funktion
0x00	No-Funktion
0x01	Port als Ausgang setzten
0x02	Port als Eingang setzten

Tabelle 6.1: Werte des Mode-Registers

Durch einen Offset zur jeweiligen Basisadresse können die Ports über das Mode-Register konfiguriert werden. Zusätzlich können die Zustände der Eingangspins über das In-Value-Register (IVR) abgefragt und die Zustände der Ausgangspins durch das Out-Value-Register (OVR) gesetzt werden. Die Offsets der entsprechenden Register, bezogen auf das jeweilige Basisregister ist in Tabelle 6.2 dargestellt:

Als Memory-Mapped-Basisadressen dienen hierfür

Offset	Register	Funktion	Name
0x04	Mode-Register	Read/Write	MR
0x08	In-Value-Register	Read-Only	IVR
0x0C	Out-Value-Register	Write-Only	OVR

Tabelle 6.2: Memory-Map des Mikrocontrollers

- Basisadresse für I/O Port A = $0xFFFF0000$
- Basisadresse für I/O Port B = $0xFFFF8000$

Wahl des Musik-Eingangs
Zur Wahl des Musikeingangs müssen die angeschlossenen Taster abgefragt werden. Hierfür müssen die entsprechenden Pins als Eingang konfiguriert sein. Anschließend kann im In-Value-Register (IVR) der Wert des Schalters abgefragt werden.

Ansteuerung des Ausgangs
Zur Steuerung des Ausgangs müssen die angeschlossenen Treiber-Endstufen angesteuert werden. Hierfür müssen die entsprechenden Pins als Ausgang konfiguriert sein. Anschließend kann der Wert zur Ansteuerung der Treiber-Endstufen in das Out-Value-Register (OVR) geschrieben werden.

A) Initialisieren Sie die Ein- und Ausgangspins für das Home-Entertainment-System in einer gemeinsamen Funktion.

```
void init ( void ) {



}
```

B) Lesen Sie den Wert der Wahl-Taster aus und geben diesen über den Rückgabewert zurück. Nehmen Sie an, dass die init-Funktion zuvor ausgeführt wurde.

```
int get_switch ( void ) {




}
```

Zusätzliches Lösungsblatt: