Attention!
Achtung!

Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die Klausur selbständig und nur unter Verwendung der erlaubten Hilfsmittel in der vorgegebenen Bearbeitungszeit bearbeiten werde. Insbesondere versichere ich, keine unerlaubte Hilfe anderer Personen in Anspruch zu nehmen und während der Klausur mit keiner anderen Person außer dem Prüfungsleiter/der Prüfungsleiterin zu kommunizieren. Mir ist bekannt, dass eine unwahre Erklärung rechtliche Folgen haben und insbesondere dazu führen kann, dass die Klausur als nicht bestanden bewertet wird. Darüber hinaus ist mir bekannt, dass der Prüfer/die Prüferin in Verdachtsfällen innerhalb der Beurteilungsfrist mündliche Nachfragen zum Stoffgebiet vornehmen kann.

Bei bewiesenen Täuschungshandlungen, worunter auch Plagiate fallen, gilt die betreffende Leistung als mit „nicht ausreichend“ (5,0) bewertet. In schwerwiegenden Fällen kann der Prüfungsausschuss die oder den Studierenden von Wiederholungsprüfungen ausschließen (§ 22 Abs. 4 RPO BA / § 21 Abs. 4 RPO MA). Des Weiteren kann ein vorsätzlicher Täuschungsversuch als Ordnungswidrigkeit mit einer Geldbuße von bis zu 50.000 Euro geahndet werden (§ 22 Abs. 6 RPO BA / § 21 Abs. 6 RPO MA). Im Falle eines mehrfachen oder sonstigen schwerwiegenden Täuschungsversuches kann der Prüfling exmatrikuliert werden (§ 22 Abs. 6 RPO BA / § 21 Abs. 6 RPO MA). Zur Feststellung von Täuschungen wird entsprechende Plagiatserkennungssoftware oder sonstige elektronische Hilfsmittel eingesetzt. Eine Studierende oder ein Studierender, der den ordnungsgemäßen Ablauf einer Prüfung stört, kann von der jeweiligen Prüferin oder dem jeweiligen Prüfer oder der oder dem Aufsichtführenden nach Abmahnung von der weiteren Teilnahme an der Prüfung ausgeschlossen werden. In diesem Fall gilt die betreffende Leistung als mit „nicht ausreichend“ (5,0) bewertet (§ 22 Abs. 5 RPO BA / § 21 Abs. 5 RPO MA). Ergänzend wird darauf hingewiesen, dass jeder Fall der Fälschung von amtlichen Dokumenten der Universität Duisburg-Essen, die zum Beweis im Rechtsverkehr geeignet und bestimmt sind, wie etwa Studierendenausweise, zur Anzeige gebracht wird.

Ich versichere hiermit, dass ich die Klausur selbständig und ohne Verwendung von Hilfsmitteln in der vorgegebenen Bearbeitungszeit bearbeiten werde. Insbesondere versichere ich, keine unerlaubte Hilfe anderer Personen in Anspruch zu nehmen und während der Klausur mit keiner anderen Person außer dem Prüfungsleiter/der Prüfungsleiterin zu kommunizieren.

Mir ist bekannt, dass eine unwahre Erklärung rechtliche Folgen haben und insbesondere dazu führen kann, dass die Klausur als nicht bestanden bewertet wird. Darüber hinaus ist mir bekannt, dass der Prüfer/die Prüferin in Verdachtsfällen innerhalb der Beurteilungsfrist mündliche Nachfragen zum Stoffgebiet vornehmen kann.

Programmieren in C

Klausur WS - 10.03.2021

- *The exam has 7 pages and 4 tasks. You need to reach 40 out of 80 points to pass the exam.*

  Die Klausur hat 7 Seiten und 4 Aufgaben. Um zu bestehen, müssen Sie insgesamt 40 von 80 Punkten erreichen.

- *You have 90 minutes to solve the exam.*

  Sie haben 90 Minuten um die Klausur zu bearbeiten.

- *Write your answers in a PDF-document. You can use any tool you prefer (e.g. LibreOffice). Clearly mark the beginning and end of each task in your PDF-documen. Answers that cannot be associated with a question will not be graded.*

  Schreiben Sie Ihre Antworten in ein PDF-Dokument. Sie können dafür ein beliebies Tool verwenden (bspw. LibreOffice). Markieren Sie deutlich Beginn und Ende jeder Aufgabe. Lösungen, die keiner Aufgabe zugeordnet werden können, werden nicht berücksichtigt.

- *This exam is an open book exam.*

  Bei dieser Klausur handelt es sich um eine Kofferklausur.

- *Cheating attempts, especially working as a group or plagiarism will result in failing the examination and further steps (as stated in the examination regulations) will be taken.*

  Täuschungsversuche insbesondere Gruppenarbeit und Plagiate führen zum Nichtbestehen der Prüfung. Ausserdem behalten wir uns vor weiter Schritte im Sinne der Klausurregelung einzuleiten.

Task	Points	Score
Fun with Files	13	
Scopes	10	
Diverse Fragen	22	
Programmieren	35	
Total:	80	

1. Task: Fun with Files [13]

(a) Abbildung 1 zeigt eine `.h`- und einen `.c`-Datei. Beschreiben Sie, was beim Kompilieren der `.c`-Dateien geschieht. Wie lässt sich ein gegebenenfalls auftretender Fehler hier verhindern. [2]

```c
#include "include.h"
#include "include.h"
```

```c
#ifndef INCLUDE

int a;

#endif //INCLUDE
```

Abbildung 1: Links: der `.c`-File, Rechts: der `.h`-File

(b) Abbildung 2 zeigt eine Quelldatei. Ist diese fehlerhaft? Beachten Sie insbesondere Zeile 1. Begründen Sie Ihre Antwort. [2]

```c
#include "Message.c"

int main(void){
    Message_send("hello world");
    return 0;
}
```

Abbildung 2: `main.c`

(c) Abbildung 3 zeigt drei Quelldateien. Das Programm verhält sich nicht wie beabsichtigt. Entgegen der Intention, gibt das print statement in Zeile 8 die Zahl 0 aus. Was ist der Grund dafür? Wie kann man das Programm korrigieren, sodass der erhöhte Wert von a ausgegeben wird. [4]

```c
#include "a.h"
// file: a.c

void increase_a() {
    a += 1;
}
```

```c
// file: a.h

static int a = 0;

void increase_a();
```

```c
#include "a.h"
#include <stdio.h>

// file: main.c

int main(void){
    increase_a();
    printf("%i\n", a);
}
```

Abbildung 3: Links: `a.c`; Mitte: `a.h`; Rechts: `main.c`

(d) Gegeben sind die Quelldateien in der Abbildung 4 mit der main-Funktion in Abbildung 5 [5]

- Zu welchem Fehler führt das Kompilieren und Linken dieser Dateien?
- Warum tritt dieser Fehler auf?
- Handelt es sich dabei um einen Compiler- oder Linker-Fehler?
- Passen Sie genau eine Zeile in `a.c` oder `a.h` an, um diesen Fehler zu beheben und 1 ausgeben zu lassen. Dabei soll keine Warnung des Compilers provoziert werden.

```h
// file: a.h
#include <stdint.h>

uint64_t a;
```

```c
// file: a.c
#include "a.h"

uint64_t a = 1;
```

Abbildung 4: Links: `a.h`; Rechts: `a.c`

```c
// file: main.c
#include "a.h"
#include <stdio.h>

int main(void){ printf("%i\n", a);}
```

Abbildung 5: `main.c`

2. Task: Scopes [10]

Folgender Code gibt mehrmals Werte auf der Konsole aus. Nennen Sie alle ausgegebenen Werte. Nutzen Sie dabei die Zeilennummern um zu markieren um welche Ausgabe es sich handelt.

```
1  #include <stdio.h>
2  #include "memory.h"
3
4  static int count = 10;
5  static int a = 500;
6
7  void firstRun(int count);
8  void secondRun(int count);
9  int f(int);
10
11 int data[] = { 0x45, 0xA4, 0x43, 0x66, 0xF3, 0xF2, 0xEE, 0x1E, 0xCF,
12                0x55, 0x55, 0x32, 0x22, 0xFE, 0xC3, 0xC5, 0xAE, 0x12,
13                0xAA, 0xBB, 0xB5, 0x6A, 0x6C, 0x6E, 0x7A, 0x8F, 0xF8,
14                0xff, 'a', 'x', '\0', 0x55, 0x12, 0x00, 0x01, 0xA1,
15                0xF1, 0xE1 };
16
17 int main(void) {
18     printf("%d\n", count);
19     firstRun(++count);
20     printf("%d\n", count--);
21     secondRun(--count);
22     printf("%d\n", count);
23 }
24
```

```
25 void firstRun(int count){
26     printf("%d\n", ++count);
27     if(--count > 10){
28         int count = 1;
29         printf("%d\n", count);
30         int *coun = malloc(f(a));
31         data[count] = count;
32         printf("%d\n", data[count]++);
33         *coun = data[count+1];
34     }
35     printf("%d\n", count);
36 }
37
38 void secondRun(int count){
39     printf("%d\n", count);
40     if(count = 10){
41         printf("%d\n", ++count);
42     }
43     printf("%d\n", ++count);
44 }
45
46 int f(int a) {
47     if (a <= 0) {
48         return 0;
49     }
50     else if (a == 1) {
51         return a;
52     }
53     else {
54         return f(a-1) + f(a-2);
55     }
56 }
```

3. Task: Diverse Fragen [22]

(a) Wie implementiert ein optimierender Compiler folgende Zeile? Begründen Sie ihre Antwort. [2]

```
*a + 10 >> 8;
```

(b) Welche Werte haben `a` und `b` nach folgender Zeile Code: [2]

```
float a; int b; a = b = 3.4;
```

(c) i. Was passiert wenn man versucht folgende Dateien zu kompilieren, zu linken und das entstehende Programm auszuführen? Begründen Sie ihre Antwort. [4]

```c
// file: main.c
#include "call.h"

void main(void){ function(1); }
```

```c
// file: call.c
#include <stdio.h>
#include "call.h"

void function(int a, int b){
    printf("%i", a);
    printf("%i", b);
}
```

```c
// file: call.h
int function(int a);
```

ii. Was geschieht stattdessen, wenn man den Header `call.h` im Source-File `call.c` inkludiert? Begründen Sie ihre Antwort. [2]

(d) Was ist modulare Programmierung in C und wofür werden dabei *incomplete data types* verwendet? [8]

(e) Der C-Standard erlaubt verschiedene Größen für enums. Daher geben einem die meisten Compiler, die Möglichkeit diese Größe beim Aufruf als Kommandozeilenparameter festzulegen. In einem Projekt wurde folgende struct-Definition im Header `FruitBasket.h` verwendet: [4]

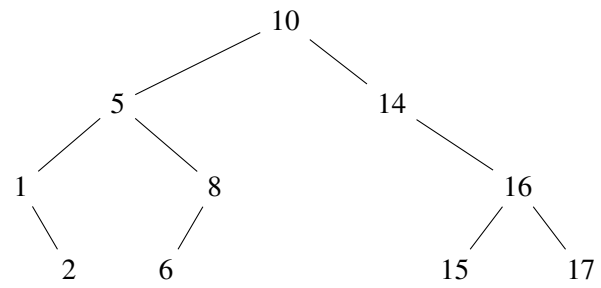
```
typedef enum {
        APPLE,
        ORANGE,
} Fruit;

typedef struct FruitBasket {
        Fruit fruits[10];
        int current_size;
} FruitBasket;
```

Welches Problem kann sich hieraus ergeben, wenn `FruitBasket.h` in verschiedenen `*.c` Dateien inkludiert wird? Wie lässt sich dieses Problem beheben?

4. Task: Programmieren [35]

In dieser Aufgabe sollen Sie einige von uns spezifizierte Funktionen implementieren. In Abbildung 6 finden Sie einige Datentypen und Funktionsdefinitionen. Ausserdem zeigt Abbildung 6 den Baum nach dem die Funktion `Tree_insert` in der hier angegebenen Reihenfolge auf jedem Element der Sequenz (10, 5, 1, 8, 14, 16, 17, 15, 2, 6) aufgerufen wurde. Beachten Sie, dass der Baum dabei immer so sortiert bleibt, dass das Datum im linken Kindknoten immer kleiner-gleich dem Datum im Elternknoten und das Datum im rechten Kindknoten immer größer ist als das Datum im Elternknoten.

Für die gesamte Aufgabe gilt: Ihre Lösung muss den von uns in Abbildung 6 gezeigten Code nutzen und mit diesem kompatibel sein.




```c
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>
#include "tree_task.h"

typedef struct Node Node;
typedef struct Tree Tree;

struct Node {
    Node *left;
    Node *right;
    int data;
};

struct Tree {
    Node *root;
    int size;
};

static void init_Node(Node *self, int data){
    self->left = NULL;
    self->right = NULL;
    self->data = data;
}

static void init_tree(Tree *self) {
    self->root = NULL;
    self->size = 0;
}

Tree *Tree_create(void){
    Tree *tree = malloc(sizeof(Tree));
    init_tree(tree);
    return tree;
}
```

Abbildung 6: Links: Datenstruktur nach dem Einfügen der Sequenz, Rechts: Definition der Datenstrukturen und Initialisierungsroutinen.

(a) Implementieren Sie die Funktion [10]

```c
bool Tree_contains(Tree *self, int data)
```

Diese soll überprüfen, ob sich das Datum `data` bereits im Baum befindet und im positiven Fall `true` zurückgeben.

(b) Schreiben Sie die Funktion [10]

```c
void Tree_print(Tree *self)
```

Diese soll die Elemente des Baums in aufsteigender Reihenfolge auf der Konsole ausgeben.

(c) Implementieren Sie die Funktion [15]

```c
void Tree_insert(Tree *self, int data)
```

Diese soll ein neues Datum in den Baum einfügen. Dabei soll die zu Beginn der Aufgabe genannte Bedingung berücksichtigt werden. Das bedeutet der Code

```
for (int i=0; i < sequence_length; i++) {
  Tree_insert(tree, sequence[i]);
}
```

wobei `sequence` die eingangs erwähnte Zahlenreihe darstellt, soll den Baum in Abbildung 6 erzeugen. Falls Sie für ihre Lösung noch andere abstrakte Datentypen wie `List`, `Queue`, `Stack`, etc. benötigen, so können Sie annehmen, dass diese implementiert vorliegen. Geben Sie in diesem Fall die von ihnen verwendeten Funktionsprototypen an. *Wichtig: Natürlich können Sie dabei* ***keine*** `Tree` *Implementierung wählen.*