Hochschule RheinMain
Studiengang Informatik - Technische Systeme
Prof. Dr. Robert Kaiser

Probeklausur Hardwarenahe Programmierung I (LV1511)
Wintersemester 2021/22

Nachname:	Vorname:
Matrikelnummer:	
Datum: X.XX.2022	**Unterschrift:**

Sie erhalten eine geheftete Klausur. **Bitte lösen Sie die Heftung nicht**. Bitte tragen Sie zu Beginn der Bearbeitungszeit Ihren Namen und Ihre Matrikelnummer an den dafür vorgesehenen Stellen ein und unterschreiben Sie die Klausur. Die Klausur ist **nur mit Unterschrift** gültig. Die Klausur muss mit dem Verlassen des Raumes abgegeben werden.

Zum Bestehen der Klausur sind 30 Punkte (50%) notwendig

Im Falle nicht ausreichenden Platzes benutzen Sie bitte zusätzliche Blätter, die Sie mit Name und Matrikelnummer versehen. Machen Sie bitte eindeutig kenntlich, auf welche Aufgabe sich Ihre Antwort bezieht.

Dauer: 60 min (Kurzklausur)
Hilfsmittel: Taschenrechner für arithmetische Operationen, eigene Formelsammlung von maximal einer doppelseitig beschriebenen DIN A4 Seite.

Punkte:

Aufgabe	**Soll-Punkte**	**Ist-Punkte**
1	12	
2	24	
3	14	
4	10	
Gesamt	60	

Note:

Aufgabe 1: (12 Punkte)

Beantworten Sie bitte folgende Fragen (je 1 P):

Frage	**Antwort**
Der Speicher eines Rechners ist in Worten zu je 32 Bit organisiert. Der Rechner hat ein Megabyte (= 2^{20} Byte) Hauptspeicher, wieviele Worte sind das?	
Wie viele verschiedene Bitmuster lassen sich in einem solchen 32-bit Wort darstellen?	
Woraus ergibt sich die Bedeutung dieser Bitmuster als -z.B.- Maschinencode oder Daten?	
Speicherzugriffe sind immer nur in ganzen Worten möglich. Wie viele Bits muss der Stackpointer des Prozessors mindestens haben, um alle Worte des Hauptspeichers adressieren zu können?	
Wozu dient der Programmzähler eines Prozessors?	
Was bewirkt die Ausführung eines Sprungbefehls beim Programmzähler?	
Was unterscheidet einen Unterprogrammaufruf (z.B. `call`-Befehl) von einem Sprungbefehl (z.B. `jump`)?	
Nennen Sie einen Vorteil der Maschinenprogrammierung	
Nennen Sie einen Nachteil der Maschinenprogrammierung	
Nennen Sie einen Vorteil der Hochsprachenprogrammierung	
Nennen Sie einen Nachteil der Hochsprachenprogrammierung gegenüber der Maschinenprogrammierung	
Nennen Sie zwei übliche Ansätze zur Implementierung einer Hochsprache	

Aufgabe 2: (24 Punkte)

Geben Sie für die im Folgenden wiedergegebenen C-Programme jeweils die erzeugten Ausgaben an. **(je 4P)**

Programm	Antwort

`int i = 1;`
`printf ("%d", i++);`
`printf ("%d", ++i);`	

`int x = 1234;`
`if (x / 100 < 30)`
`  printf("X");`
`else`
`  printf("Y");`
`if (x % 100 < 30)`
`  printf("!");`
`else if (x % 10 < 3)`
`  printf("*");`
`else`
`  printf("+");`	

`switch ((int) 3.3)`
`{`
`  case 3:`
`  printf("Alpha\n");`
`  case '3':`
`  printf("Beta\n");`
`  default:`
`  printf("Gamma\n");`
`}`	

`int i = 0;`
`for (int j = 5; i <= j; j--)`
`  i++;`
`printf ("%d", i);`	

```
#define MAX_CHAR ((1<<7)-1)

for (char c = 126; c <= MAX_CHAR; c++)
    printf ("%d - %x\n", c, c & 0xFF);
```

Hinweis: geben Sie nur die ersten vier Zeilen an.

Was fällt bei dem obigen Programm auf? Erklären Sie! **(4P)**

Aufgabe 3: (14 Punkte)

Gegeben sei folgendes C-Programm `program.c`.

```
int main()
{
    int i;
    printf("Hier kommt das Siebener-Einmaleins:\n");
    for(i = 1; i <= 10; i++)
        printf("%d x 7 = %d\n", i, 7*i);
    return 0;
}
```

a) Dieses Programm wird übersetzt mit:

```
gcc -Wall -o program program.c
```

Gibt es dabei ...

- Fehler, oder
- Compilerwarnungen? **(2P)**

b) Wird ein ausführbares Programm erzeugt? Falls ja: Geben Sie die Ausgabe des Programms an. Falls nein: Begründen Sie, warum kein Programm erzeugt werden konnte.**(4P)**

c) Was müssen Sie zum Programm hinzufügen, um es Fehler- und Warnungsfrei übersetzen zu können? Was wird damit erreicht?**(2P)**

d) Ist `printf()` Bestandteil der Sprache C[1]? Woher wird die **Implementierung** der Funktion `printf()` bezogen?**(2P)**

e) Geben Sie ein `Makefile` zur Übersetzung des Programmes an. Gestalten Sie das `Makefile` so, dass die oben verwendeten Optionen (`-Wall`) zur Anwendung kommen und dass Sie leicht statt des Standardcompilers einen anderen Compiler (z.B. `avr-gcc`) verwenden können.**(4P)**

[1]D.h.: ist `printf` ein reserviertes Wort, so wie `while`, `sizeof` oder `return`?

Aufgabe 4: (10 Punkte)

Die Standardbibliothek definiert die Funktion `qsort()`[2] folgendermaßen:

```
#include <stdlib.h>

void qsort(void *basis, unsigned int anzahl,
           unsigned int groesse,
           int (*vergleich)(void *, void *));
```

In der Manualseite heisst es dazu:

Die Funktion ***qsort()*** *sortiert ein Array mit* ***anzahl*** *Elementen der Größe* ***groesse****. Das Argument* ***basis*** *zeigt auf das erste Element des Arrays.*

Es werden jeweils zwei Zeiger auf miteinander zu vergleichende Elemente des Arrays an ***vergleich*** *übergeben.* ***vergleich*** *muss eine Ganzzahl zurückgeben, die kleiner, gleich oder größer Null ist, je nachdem, ob das erste Argument kleiner, gleich oder größer als das zweite ist.*

a) Was für einen Typ hat der Funktionsparameter `vergleich`? **(2P)**

b) Sowohl das Argument `basis` als auch die beiden Argumente für `vergleich` sind vom Typ `void *`. Was für Objekte sind das und wieso wird hier nicht einfach der Typ der Arrayelemente verwendet?**(3P)**

c) Welche Bedeutung hat das „`void`“ , das vor `qsort` steht? **(1P)**

[2]Es handelt sich hier um den so genannten „Quick Sort“ Algorithmus.

d) Das folgende Programm liest eine Anzahl Ganzzahlen ein und sortiert sie mittels `qsort()` in aufsteigender Reihenfolge:

```
#include <stdio.h>
#include <stdlib.h>

int vergl(void *a, void *b)
{









}


int main()
{
    int daten[100];
    int i, n;
    for(i = 0; i < 100; i++)
    {
        printf("Gib die %d-te Zahl ein: ", i);
        if(scanf("%d", &daten[i]) != 1)
            break;
    }
    qsort((void *)&daten[0], i, sizeof(int), vergl);
    for(n = 0; n < i; n++)
        printf("%d-te Zahl: %d\n", n, daten[n]);
}
```

Geben Sie den Inhalt der Funktion `vergl()` an.**(4P)**
Hinweis: Tragen Sie den Code im obigen Listing ein.