

KLAUSUR im FACH *Ingenieur-Informatik* im Wintersemester 2019/2020

Name, Vorname:

Studiengang/Semester:

Prüfer: Dipl.-Ing. (FH) Sebastian Balz, Dipl.-Ing. (FH) Marko Weber, Dipl.-Ing. (FH) Rüdiger Ehret,
Prof. Dr.-Ing. Daniel Fischer

Bearbeitungshinweise

1. Tragen Sie auf jeder Seite in der Kopfzeile Ihre Matrikelnummer ein.
2. Der Aufgabensatz (inkl. Deckblatt und Anhang), der aus 16 Seiten besteht (Seite 1 bis 16), ist auf Vollständigkeit zu überprüfen.
3. Der Aufgabensatz ist mit den Lösungsblättern abzugeben.
4. Lösungen auf selber mitgebrachten Lösungsblättern werden nicht ausgewertet. Verwenden Sie die Ihnen ausgeteilten Lösungsblätter und tragen Sie auch dort Ihre Matrikelnummer ein.
5. Bei Rechenaufgaben muss der Lösungsweg ersichtlich und lesbar sein, sonst erfolgt keine Bewertung der Aufgabe oder des Aufgabenteils.
6. Die Bearbeitungszeit beträgt 90 Minuten.
7. Es wird hiermit darauf hingewiesen, dass vom Prüfungsamt nicht vorher geprüft wurde, ob Sie das Recht bzw. die Pflicht zur Teilnahme an dieser Klausur haben. Die Teilnahme erfolgt auf eigene Gefahr, gleichzeitig bekundet die Teilnahme die Zustimmung zu diesem Passus.
8. Hilfsmittel:
 - C-Coding Styleguide (ist selbst mitzubringen) – nur Markierungen mit einem Marker sind erlaubt (keine eigenen Notizen)
 - ASCII-Tabelle und Kurzreferenz Bibliotheksfunktionen (wird ausgeteilt)
 - Taschenrechner sind **nicht** erlaubt – auch **keine** Smartphones

9. Die Nichteinhaltung des C-Coding Styleguides führt zu Punktabzug**10. Bewertung:**

Gesamtpunktzahl: 100 Punkte (10% Überhang)

Note 1,0: 90 Punkte

Note 4,0: 45 Punkte

Aufgabe	1	2	3	4	5	6	7	SUMME	
Punkte	10	10	20	10	10	15	25	100	NOTE
Erreichte Punkte									

Aufgabe 1:**(10 Punkte)****1.1** Der Linker erhält den Source- und den Objectcode als Input. (2 Punkte)☐ Ja☐ Nein

Begründung:

1.2 Alle Berechnungen mit PI (z.B. `dAreaCircle = 2*PI*dr*dr;`) sind korrekt.`#define PI 3,1415`☐ Ja☐ Nein

Begründung:

1.3 Bei folgender Programmzeile handelt es sich um eine Definition. (2 Punkte)`enum StudyProgram {EI, EI+, MKA, MK+, EP, EP+, EI3nat, AI};`☐ Ja☐ Nein

Begründung:

1.4 Die Funktion scanf ist eine unsichere Funktion. (2 Punkte)☐ Ja☐ Nein

Begründung:

1.5 In einem Struktogramm (Nassi-Shneiderman-Diagramm) gibt es **keinen** Unterschied zwischen einer for- und einer do-while-Schleife. (2 Punkte)☐ Ja☐ Nein

Begründung:

Ggf. hier noch Skizze:

Aufgabe 2: (10 Punkte)

Schreiben Sie hinter die Kommentare, was *exakt* in der Konsole auf einem **32-Bit System (x86)** ausgegeben wird. Im Kommentar sind die möglichen Punkte aufgeführt.

Hinweis: Hexadezimale Ziffer A-F werden bei printf mit dem Formattierer **%x** als **Kleinbuchstaben** ausgegeben. **Führende Nullen** werden durch **%x** auch **unterdrückt**.

```
#include "MyVars.h"
```

```
void AG2(void)
{
    unsigned char uc1;
    unsigned char uc2;

    struct Flags sFlags;
    char acArray[10];
    char* pcacArray = &acArray[3];
    strcpy(acArray, "123");

    printf("%d\n", *pcacArray);                // _____ (1P)

    printf("%d\n", strlen(acArray)-sizeof(acArray)); // _____ (1P)

    printf("%u\n", sizeof(struct ConvertManager)); // _____ (1P)

    printf("%u\n", sizeof(sFlags));            // _____ (1P)

    printf("%d\n", Q(acArray[1],acArray[2])); // _____ (2P)

    uc1 = 0x0F;
    printf("%x\n", ((uc1>>5U)<<3U));          // _____ (1P)

    uc1 = 0xF0;
    uc2 = 0x7B;
    printf("%x\n", ((~uc1)&uc2));              // _____ (1P)

    uc1 = 0x0F;
    uc2 = 0x0B;
    uc1 ^= (--uc2);
    printf("%x\n",uc1 );                     // _____ (2P)
}
```

```
//MyVars.h
#pragma once

//Ternary Operator should not be used!!!
#define Q(a,b) (((a)<(b)) ? (a) : (b))

struct Flags
{
    unsigned int uiVal1 : 16;
    unsigned int uiVal2 : 14;
};

struct ConvertManager
{
    union uV
    {
        struct Flags* apbf[5];
        short int asiValues[5];
    }uValues;
};
```

Aufgabe 3:**(20 Punkte)****3.1** Wandeln Sie die folgenden Zahlen in ein anderes Zahlensystem um. (3 Punkte)

$$123_{20} = \underline{\hspace{4cm}}_{10}$$

$$1011\ 1010\ 1101\ 1100\ 1010\ 1111\ 1110_2 = \underline{\hspace{4cm}}_{16}$$

$$A74_{16} = \underline{\hspace{4cm}}_8$$

3.2 [C2] Implementieren Sie in C eine Funktion `foo` mit **genau einem** `if` und **keiner** Schleife. Die Funktion soll $v(G) = 3$ haben. ($v(G)$ ist zyklomatische Komplexität) (2 Punkte)

```
int foo(int iA, int iB, int iC)
{
    int iRet;
```

```
        return iRet;
}
```

3.3 [C1] Implementieren Sie ein Funktionsmakro, welches das Byte 1 einer 32-Bit Zahl zurückgibt. (3 Punkte)

Byte 3	Byte 2	Byte 1	Byte 0
--------	--------	--------	--------

3.4 Der folgende Programmcode enthält zwei Fehler in zwei unterschiedlichen Zeilen. Korrigieren Sie diese Fehler! Jeweils 2 Punkte Abzug bei falscher Korrektur. (4 Punkte)

```
#include <stdio.h>

int main(void)
{
    FILE* pf;
    char acLine[255];
    char acFilename[] = "c:\\temp\\students.txt";
    pf = fopen(acFilename,"r");

    if (pf != NULL)
    {
        while (feof(pf) == 0)
        {
            fgets(acLine, 255, pf);
            printf("%s",acLine);
        }
        fclose(FILE);
    }
    return 0;
}
```

3.5 [C1] (8 Punkte)

3.5.1 Deklarieren Sie einen komplexen Datentyp, welcher die Typenbezeichnung (type) als Zeichenkette, das Gewicht (weight) in Gramm und den Verkaufspreis (price) eines Smartphones enthält. Berücksichtigen Sie den C-Coding Styleguide. (1 Punkt)

3.5.2 Allokieren Sie sich 100 Variablen als Array dieses Typs (3.5.1) einmal statisch und einmal dynamisch! (2 Punkte)

Statisch:

Dynamisch:

3.5.3 Weisen Sie dem jeweils letzten Element der beiden Arrays den Wert 200U für das Gewicht (weight) zu! (2 Punkte)

3.5.4 Deklarieren Sie eine Funktion mit dem Namen foo, die eine Variable aus 3.5.1 per Value übergeben bekommt. (1 Punkt)

```
void foo(                                     );
```

3.5.5 Rufen Sie nun die Funktion foo zweimal auf. Übergeben Sie jeweils das letzte Element der beiden Arrays.

```
foo(                                     ); //Last element static allocated
```

```
foo(                                     ); //Last element dynamic allocated
```

Aufgabe 4: [C1]**(10 Punkte)**

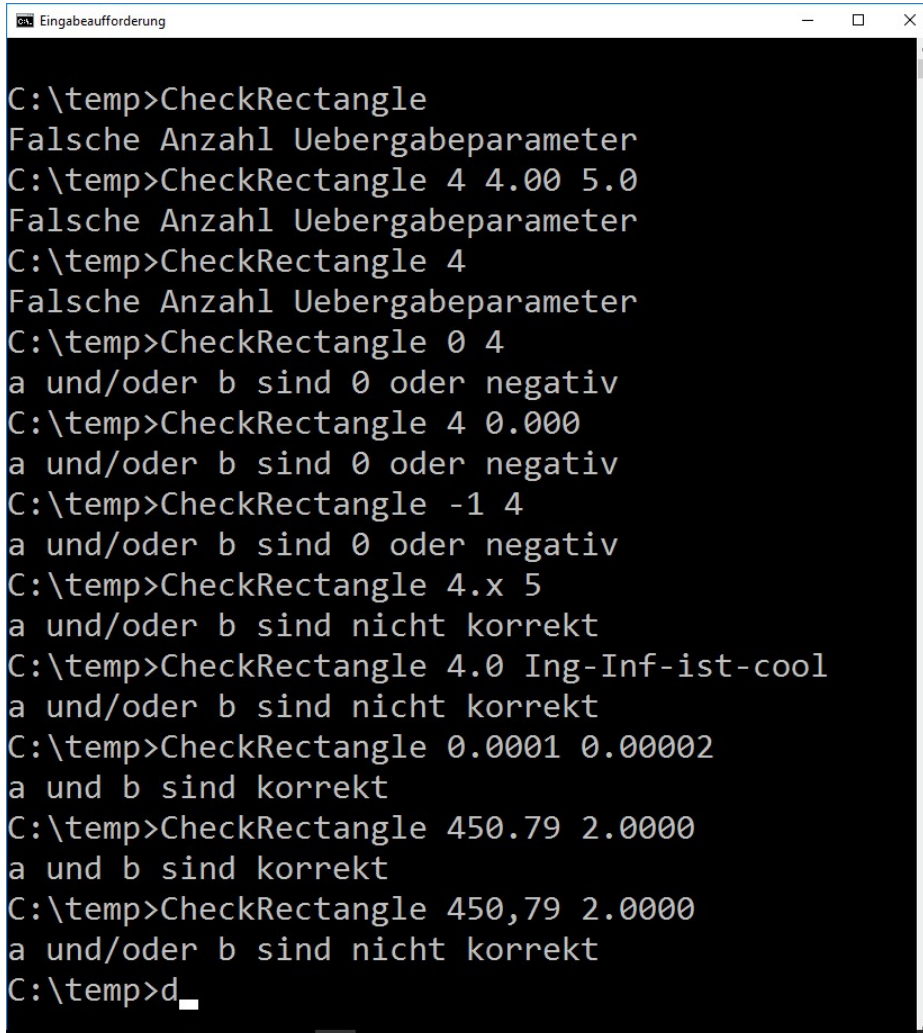
Bestimmen Sie in C die Werte für x, y und z, sodass diese die folgende Gleichung lösen und geben Sie die Werte von x, y und z noch auf der Konsole aus.

$$3 \cdot x + 7 \cdot y = -97 + 11 \cdot z \qquad x, y, z \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

Aufgabe 5: [C3]**(10 Punkte)**

Implementieren Sie ein Programm `CheckRectangle` in C, welches mittels Übergabeparameter beim Aufruf die Seitenlängen `a` und `b` eines Rechtecks übergeben bekommt. Geben Sie auf der Konsole aus, ob die übergebenen Seitenlängen korrekt sind.

Details entnehmen Sie aus dem folgenden Screenshot.



```
C:\temp>CheckRectangle
Falsche Anzahl Uebergabeparameter
C:\temp>CheckRectangle 4 4.00 5.0
Falsche Anzahl Uebergabeparameter
C:\temp>CheckRectangle 4
Falsche Anzahl Uebergabeparameter
C:\temp>CheckRectangle 0 4
a und/oder b sind 0 oder negativ
C:\temp>CheckRectangle 4 0.000
a und/oder b sind 0 oder negativ
C:\temp>CheckRectangle -1 4
a und/oder b sind 0 oder negativ
C:\temp>CheckRectangle 4.x 5
a und/oder b sind nicht korrekt
C:\temp>CheckRectangle 4.0 Ing-Inf-ist-cool
a und/oder b sind nicht korrekt
C:\temp>CheckRectangle 0.0001 0.00002
a und b sind korrekt
C:\temp>CheckRectangle 450.79 2.0000
a und b sind korrekt
C:\temp>CheckRectangle 450,79 2.0000
a und/oder b sind nicht korrekt
C:\temp>d_
```

Verwenden Sie zur Lösung den ausgeteilten Lösungsbogen.

Aufgabe 6: [C2]**(15 Punkte)**

Implementieren Sie in C die Funktion `int memcmp2(void* pv1, void* pv2, size_t uiN)` ohne dabei Funktionen aus der Standardbibliothek zu verwenden.

`memcmp`: **m**emory **c**ompare

Return value	Beschreibung
<0	Das erste unterschiedliche Byte hat in pv1 einen kleineren Wert als in pv2.
=0	Speicherbereich ist gleich oder mindestens ein Nullzeiger wurde übergeben
>0	Das erste unterschiedliche Byte hat in pv1 einen größeren Wert als in pv2.

Hinweis: Nullzeiger **müssen** überprüft werden.

```
int memcmp2(const void* pv1, const void* pv2, size_t uiN)
{
```

```
}
```

Aufgabe 7: [C2]**(25 Punkte)**

Mit der Funktion **void _gotoxy(5,5)** kann der Cursor auf der Konsole an eine bestimmte Stelle (5,5) gesetzt werden. Mit **putChar('*')** können Sie dann an der Position (5,5) ein * setzen. Siehe **Anhang C!**

Implementieren Sie in C eine **rekursive** Funktion DrawLine, welche rekursiv unter Nutzung von _gotoxy und eine putChar eine Linie aus * von (usX1,usY1) nach (usX2,usY2) zeichnet. Überlegen Sie sich erst anhand der Skizze das Abbruchkriterium! Die Größe der Konsole muss nicht berücksichtigt werden.

```
void DrawLine(unsigned short usX1, unsigned short usY1,unsigned short usX2, unsigned short usY2)
{
```

```
} //Viel Erfolg!
```

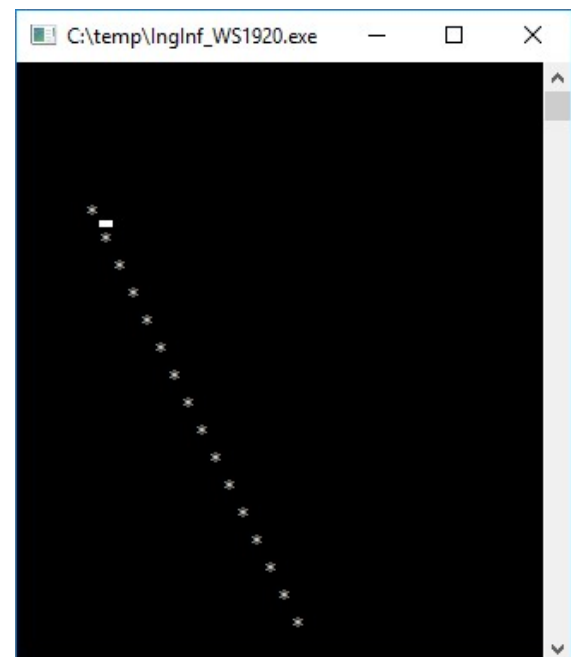
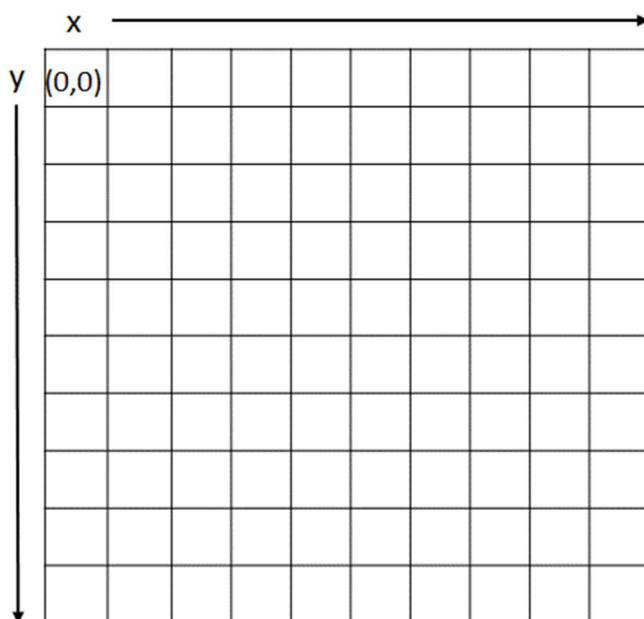
Anhang A: ASCII-Tabelle

Dez.	Hex.	Zeichen	Dez.	Hex.	Zeichen	Dez.	Hex.	Zeichen	Dez.	Hex.	Zeichen
0	0	NUL	32	20		64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	\$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27	'	71	47	G	103	67	g
8	8	BS	40	28	(72	48	H	104	68	h
9	9	HT	41	29)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	
29	1D	GS	61	3D	=	93	5D]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Anhang B: Codierung

hexadezimal	dezimal	binär
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Anhang C: Anhang zu Aufgabe 7



Bsp. für DrawLine(20,20,5,5)

Anhang D: Standardbibliotheken**ctype.h**

```
int isalnum(int iX);
int isalpha(int iX);
int iscntrl(int iX);
int isdigit(int iX);
int isgraph(int iX);
int islower(int iX);
int isprint(int iX);
int ispunct(int iX);
int isspace(int iX);
int isupper(int iX);
int isxdigit(int iX);
int tolower(int iX);
int toupper(int iX);
```

string.h

```
void* memcpy(void* pvS1, const void* pvS2, size_t uiN);
void* memmove(void* pvS1, const void* pvS2, size_t uiN);
char* strcpy(char* pcS1, char* pcS2);
char* strncpy(char* pcS1, char* pcS2, size_t uiN);
char* strcat(char* pcS1, const char* pcS2);
char* strncat(char* pcS1, const char* pcS2, size_t uiN);
int memcmp(const void* pvS1, const void* pvS2, size_t uiN);
int strcmp(const char* pcS1, const char* pcS2);
int strnmp(const char* pcS1, const char* pcS2, size_t uiN);
void* memchr(const void* pvS, int iC, size_t uiN);
char* strchr(const char* pcS, int c);
size_t strspn(const char* pcS1, char* pcS2);
char* strpbrk(const char* pcS1, const char* pcS2);
char* strrchr(const char* pcS, int iX);
size_t strspn(const char* pcS1, const char* pcS2);
const char* strstr(const char* pcS1, const char* pcS2);
void* memset(void* pvM, int iC, size_t uiN);
size_t strlen(const char* pcS);
```

```
char* strtok(char* pcStr, const char* pccDelimiters);
char* strlwr(char* pcStr); // converts string to lowercase – no C Standard
char*strupr(char* pcStr); // converts string to uppercase – no C Standard
```

stdio.h

```
int fflush(FILE* pxFile);
size_t fread(void* pvData, size_t uiSize, size_t uiNumber, FILE* pxFile);
size_t fwrite(void* pvData, size_t uiSize, size_t uiNumber, FILE* pxFile);
int fseek(FILE* pxFile, long lOffset, int iPos);
long ftell(FILE* pxFile);
void rewind(FILE* pxFile);
int feof(FILE* pxFile); //return not 0 if EOF is reached
int fputc(const char* pcS, FILE* pxFile); //return EOF if error, otherwise position
int rename(char* pcFilenameOld, char* pcFilenameNew); // return 0 if success

FILE* fopen(const char* pccFilename, const char* pccModus);
int fclose(FILE* pFile); //0 if okay, EOF if Error
int printf(const char * pccFormat, ... );
int sprintf(char* pcStr, const char * pccFormat, ... );
int scanf(const char* pccFormat, ... );
int sscanf(char* pcStr, const char* pccFormat, ... )
```

math.h

```
double acos(double dX);
double asin(double dX);
double atan(double dX);
double atan2(double dX, double dY);
double cos(double dX);
double sin(double dX);
double tan(double dX);
double cosh(double dX);
double sinh(double dX);
double tanh(double dX);
double exp(double dX);
double log(double dX);
double log10(double dX);
double pow(double dX, double dY); //xy
double sqrt(double dX);
double ceil(double dX); // Ganzzahliger Wert durch Aufrunden
double floor(double dX); // Ganzzahliger Wert durch Abrunden
double fmod(double dX, double dY); // Rest der (ganzzahligen) Division der beiden Param.

double fabs(double dX); // C99
float fabsf(float dX); // C99
long double fabsl(long double dX); // C99
```

stdlib.h

```
double atof(const char* pccValue);
int atoi(const char* pccValue);
long atol(const char* pccValue);
double strtod(const char* pccValue, char** ppcEndConversion);
long strtol(const char* pccValue, char** ppcEndConversion, int iBase);
unsigned long strtoul(const char* pccValue, char** ppcEndConversion, int iBase);
int rand(void);
void srand(unsigned int uiStartValue);
int abs(int iValue);
int labs(long int liValue);

char* itoa(int iValue, char* pcStr, int iBase);
```

time.h

Datenstrukturen

```
struct tm
{
int tm_sec;
int tm_min;
int tm_hour;
int tm_mday;
int tm_mon;
int tm_year;
int tm_wday;
int tm_yday;
int tm_isdst;
};
typedef long clock_t;
typedef long time_t

char* asctime(const struct tm* pcsTime);
time_t mktime(struct tm* psTime);
struct tm* localtime(const time_t* pcxTime);
clock_t clock(void);
time_t time(time_t* pxTime);
char* ctime(const time_t* pcxTime);

double difftime(time_t xEndtime, time_t Begintime);
```