KLAUSUR im FACH ***Ingenieur-Informatik***

im Sommersemester 2016

Name/Vorname……..

Semester……..Studiengang……..

Prüfer:
Dipl.-Ing. (FH) Rüdiger Ehret

Bearbeitungshinweise

1. Tragen Sie auf jeder Seite in der Kopfzeile Ihre Matrikelnummer ein.

2. Der Aufgabensatz (inkl. Deckblatt und Anhang), der aus 10 Seiten besteht (Seite 1 bis 10), 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:
   - handgeschriebene Formelsammlung (1 DIN A4 Blatt)

9. Bewertung:

   Gesamtpunktzahl = 100 Punkte

   Note 1,0 = 90 Punkte
   Note 4,0 = 45 Punkte

Aufgabe	**1**	**2**	**3**	**4**	**5**	**6**	**SUMME**	
Punkte	**20**	**10**	**20**	**10**	**20**	**20**	**100**	**NOTE**
Erreichte Punkte								

Aufgabe 1: (20 Punkte)
Es ist jeweils eine Antwort richtig. Fehlerhafte Antworten führen **nicht** zum Punktabzug!

1.1 Die Microsoft IDE Visual Studio verwendet zum Übersetzen einen
a) Translator
b) Interpreter
c) Compiler
d) Transformator
e) Keine Antwort (a-d) ist richtig

1.2 Wie viele Bits enthält ein Nibble?
a) 1
b) 8
c) 16
d) 256
e) Keine Antwort (a-d) ist richtig

1.3 Was ergibt 2^{16}?
a) 32
b) 256
c) 1600
d) 65536
e) Keine Antwort (a-d) ist richtig

1.4 Welcher Teil der IDE fügt die Objectcodes zusammen?
a) Linker
b) Optimierer
c) Binder
d) Compiler
e) Keine Antwort (a-d) ist richtig

1.5 Welche Bibliothek gehört nicht zum ANSI-Standard?
a) stdlib.h
b) string.h
c) conio.h
d) stdio.h
e) Keine Antwort (a-d) ist richtig

1.6 Was ist eine Präprozessor-Anweisung?
a) a=1;
b) int main(void)
c) if
d) while
e) Keine Antwort (a-d) ist richtig

1.7 Was ist der Bereich, den man mit einer signed char Variablen darstellen kann?
a) -128..128
b) -127..127
c) 0..256
d) -127..128
e) Keine Antwort (a-d) ist richtig

1.8 Was liefert sizeof(double) zurück (Visual Studio, BWLehrpool)?
a) 32
b) 2
c) 4
d) 8
e) Keine Antwort (a-d) ist richtig

1.9 Der ASCII-Code des End of String Zeichens ist (Dezimal):
a) '\0' bzw. 0x00
b) 'E'
c) 255
d) -1
e) Keine Antwort (a-d) ist richtig

1.10 Wird einer Integervariablen i1 der Wert einer Fließkommavariablen f1 zugewiesen (i1=f1;) so handelt es sich um einen
a) Zuweisungscast
b) Compilercast
c) impliziten Cast
d) expliziten Cast
e) Keine Antwort (a-d) ist richtig

1.11 Was ist bezüglich Strukturierter Programmierung nicht erlaubt?
a) Schleife in einer Schleife
b) Abfrage in einer Abfrage
c) zwei return in einer Funktion
d) do-while-Schleife mit zwei Bedingungen
e) Keine Antwort (a-d) ist richtig

1.12 Was ist im Zusammenhang mit dem Schlüsselwort const bei der Definition einer Konstanten (konstante „Variable") zu beachten?
a) Die Variable befindet sich im Code Segment
b) Es ist immer nur mit einem typedef erlaubt
c) Es hat immer bei der Definition auch eine Wertzuweisung zu erfolgen
d) Die Variable darf später nur bei Multiplikationen und nie bei Divisionen verwendet werden.
e) Keine Antwort (a-d) ist richtig

1.13 Statische lokale Variablen befinden sich im folgenden Speicherbereich:
a) Stack (Stack Segment)
b) Heap (Heap Segment)
c) Globale Variablen (Daten Segment)
d) Programm Code (Code Segment)
e) Keine Antwort (a-d) ist richtig

1.14 Was liefert malloc als Rückgabewert?
a) int
b) int*
c) void
d) void*
e) Keine Antwort (a-d) ist richtig

1.15 Beim Anlegen eines neuen Projektes ist folgender Warning Level eingestellt:
a) 1
b) 2
c) 3
d) 4
e) 8

1.16 Was ergibt sich aus einem Structure Chart?
a) Die notwendigen Bibliotheken
b) Nur die Funktionsnamen
c) Das Flußdiagramm
d) Die Funktionsdeklarationen
e) Keine Antwort (a-d) ist richtig

1.17 Was ist die korrekte Definition eines typenlosen Zeigers (generischer Zeiger)?
a) int * generic;
b) void (*generic) ( );
c) void * generic;
d) generic * void;
e) Keine Antwort (a-d) ist richtig

1.18 Welches ist die korrekte Definition eines Funktionszeigers, der auf die Funktion atoi zeigen soll?
a) int (*fp)(const char*);
b) double *fp(const void);
c) void (*fp)(const void);
d) void *fp (const char*);
e) Keine Antwort (a-d) ist richtig

1.19 Was unterscheidet die Funktion printf grundsätzlich von fopen?
a) Die Anzahl der Übergabeparameter ist bei printf nicht konstant
b) printf hat keinen Rückgabewert
c) fopen gehört nicht zum ANSI-Standard
d) Nur printf erhält eine Zeichenkette als Übergabewert
e) Keine Antwort (a-d) ist richtig

1.20 Wie viel Speicher benötigt das folgende Array (Visual Studio, BWLehrpool):
unsigned short int Cube [4][3][2];
a) 12
b) 9
c) 24
d) 48
e) 96
f) Keine Antwort (a-e) ist richtig

Aufgabe 2: (10 Punkte)
2.1 Wandeln Sie die folgenden Zahlen in das andere Zahlensystem um: (4 x 1 Punkt)

$1001\ 0110\ 1010_2 =$ **4552_8**
$1001\ 0110\ 1010_2 =$ **$96A_{16}$**
$13_{11} =$ **14_{10}**
$17_{10} =$ **11_{16}**

2.2

Allokieren Sie dynamisch in C hundert Variablen vom Typ struct Employee und weisen Sie aufsteigend für empno die Werte 1-100 zu. (Erste Variable 1, letzte 100) (6 Punkte)	`struct Employee`
	`{`
	`  unsigned int empno;`
	`  char name [100];`
	`};`

```
// Nur bei vollständiger Korrektheit Teilpunkte 1 P, 2 P und 3 P

  struct Employee * pe; // 1 P

  pe = (struct Employee*) malloc(100 * sizeof(struct Employee));
// 2 P – (struct Employee*) ist in C optional, d.h. nicht notwendig


  for (i = 0; i < 100; i++)
  {
    (pe + i)->empno = i + 1;
     // andere Schreibweise auch ok: pe[i].empno
  }
// 3 P
```

Matrikelnummer

Aufgabe 3: (20 Punkte)

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

MyVars.h

```
union County
{
    char cFrance;
    char cGermany;
};
struct Motor
{
    union County uCounty[4];
    char* pcName;
    int   iPower[2];
    float fCostsInternal;
};
#define SIZE 25
```

```
void printit(void)
{
  enum semester { MK, MKplus, EP, EPplus, EI, EIplus, EI3nat };
  enum semester esem1 = EIplus;
  enum semester esem2 = MKplus;
  struct Motor mymotor;
  unsigned char uc1, uc2;

  mymotor.uCounty[2].cFrance = 'F';
  mymotor.uCounty[2].cGermany = 'G';
  mymotor.pcName = (char*) malloc(SIZE * 2 * sizeof(char) );
  strcpy(mymotor.pcName,"Power Engine - ALPHA");
  mymotor.iPower[0] = 10;
  mymotor.iPower[1] = 20;
  mymotor.fCostsInternal = 555.5f;

  printf("%d\n", sizeof(int));                // 4_____________ 1 P
  printf("%d\n", sizeof(int*));               // 4_____________ 1 P
  printf("%d\n", sizeof(esem1));              // 4_____________ 1 P
  printf("%d\n", sizeof(union County));       // 1_____________ 1 P
  printf("%d\n", sizeof(mymotor));            // 20____________ 1 P
  printf("%d\n", sizeof(mymotor.pcName));     // 4_____________ 1 P
  printf("%d\n", strlen(mymotor.pcName));     // 20____________ 1 P
  printf("%d\n", sizeof(struct Motor*));      // 4_____________ 1 P
  printf("%d\n", (esem2 + esem1));            // 6_____________ 1 P
  printf("%c\n", mymotor.uCounty[2].cFrance); // G_____________ 1 P

  printf("%s\n", strtok(mymotor.pcName,"- ")); // Power_________ 2 P
  printf("%d\n", (mymotor.pcName[0]-mymotor.pcName[6]));   // 11 2 P
  printf("%d\n", (&mymotor.iPower[0]-&mymotor.iPower[1])); // -1 2 P

  uc1 = 0xF2;
  uc2 = 0x0F;
  printf("%x\n",(uc1 | (uc2 << 2)));          // fe____________ 2 P
  printf("%x\n",((uc1 & uc2) ^ 0xFF));        // fd____________ 2 P
}
```

Aufgabe 4: (10 Punkte)
Gegeben sei der folgende Programmcode in C. Dieser enthält einen Fehler! Markieren Sie diesen Fehler und erklären Sie, warum es sich um einen Fehler handelt.

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

int main(int argc, char* argv[])
{
  int i;

  for(i=1; i <= argc; i++)
  {
    if (!strcmp(argv[i], "-enable-this"))
       printf("'this' aktiviert.\n");
    else
       if(!strcmp(argv[i], "-enable-that"))
          printf("'that' aktiviert.\n");
       else
          printf("Datei verwenden: %s\n", argv[i]);
  }

 return 0;
}
```

Das erste Argument (Index 0) ist der Programmname (exe-Datei und Pfad). Mögliche weitere Argumente, die über die Kommandozeile eingegeben werden, folgen darauf.
Die Variable i darf in der for-Schleife nie > i-1 werden, da sonst außerhalb auf des Zeigerarray argv zugegriffen wird (argc ist Anzahl der Argumente, Index ist im Bereich 0 bis (argc-1));

Aufgabe 5: (20 Punkte)
Implementieren Sie die Funktion reverse_string, welche die Reihenfolge der Zeichen der als char-Zeiger übergebenen Zeichenkette dreht (reverse).
Beispiel:
Zeichenkette p1 vor dem Aufruf: „Peter“
Zeichenkette p1 nach dem Aufruf: „reteP“

Vorgaben: Es dürfen keine Bibliotheksfunktionen und keine weiteren Variablen verwendet werden. Sonst wird die Aufgabe mit 0 Punkten bewertet!

```
void reverse_string (char * p1)
{
  char * p2;
  char ch;

  if (*p1 != 0x00)  // 2 P
  {
    p2 = p1; // 2 P

    while (*(p2 + 1) != 0x00)  // 6 P
      p2++;

    while (p2 > p1)  // 10 P
    {
      ch = *p2;
      *p2 = *p1;
      *p1 = ch;
      p1++;
      p2--;
    }

}
```

Aufgabe 6: (20 Punkte)
Implementieren Sie die Bibliotheksfunktion myitoa (itoa: integer to ASCII), welche einen übergebenen Wert value zur Basis base in eine Zeichenkette str umwandelt. Es dürfen keine Bibliotheksfunktionen verwendet werden. Nehmen Sie an, dass base >1 ist und der Speicher auf den str zeigt, genügend groß ist.
Wichtig: myitoa verwendet reverse_string aus Aufgabe 5. Sie können diese Aufgabe aber auch unabhängig von Aufgabe 5 lösen.

```
char* myitoa (unsigned int value, char * str, unsigned int base)
{
  unsigned int rest;
  char * p = str;                 // 2 P

  while (value > 0)
  {
    rest = value % base;          // 4 P
    value = value / base;         // 4 P
    if (rest < 10)
       *p = rest + 48;            // 6 P
    else
      *p = rest + 55;

    p++;                          // 2 P
  }

  *p = 0x00;                      // 2 P

reverse_string(str);
return str;
}
```

Viel Erfolg!

Anhang: 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