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

im Wintersemester 2014/2015

Semester……..Studiengang……..

Prüfer:
Dipl.-Ing. (FH) Sebastian Balz, Dipl.-Ing. (FH) Sebastian Stelzl, 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 = 40 Punkte

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

Aufgabe 1: **20 Punkte**

Es ist jeweils eine Antwort richtig. **Fehlerhafte Antworten führen zu Punktabzug bei dieser Aufgabe! Es sind aber keine negativen Punkte für diese Aufgabe möglich.**

1.1 C ist eine
a) Dokumentenorientierte Sprache
b) Höhere Programmiersprache
c) Assemblersprache
d) Maschinensprache

1.2 C89 bedeutet, dass
a) es sich um den C-Standard von 1989 handelt
b) es sich um die C-Version mit den 89 Schlüsselworten handelt
c) C 1989 erfunden wurde
d) 89% von C standardisiert sind

1.3 Welche Anwendungen werden heutzutage **nicht** mehr in C programmiert?
a) Echtzeitsysteme
b) Treiberprogramme
c) Web-Anwendungen
d) Embedded Systems

1.4 Welche Bibliotheksfunktion hat eine variable Anzahl an Übergabeparametern (formalen Parametern)?
a) printf
b) strcpy
c) strncpy
d) flexlm

1.5 Mit welcher Bibliotheksfunktion kann man abfragen, ob sich Zeichen im Keyboardbuffer befinden? Dabei soll nichts eingelesen werden!
a) _getch
b) scanf
c) _kbhit
d) getchar

1.6 Welches Schlüsselwort von C benötigt man nicht, da es als „default“ (=voreingestellt) gilt?
a) auto
b) static
c) unsigned
d) if

1.7 Was ist der Bereich, den man mit einer signed char Variablen darstellen kann?
a) -128..128
b) -127..127
c) -128..127
d) -127..128

1.8 Wie weist man einer unsigned char Variablen a das Zeichen A zu?
a) a ="A";
b) a = A;
c) strcpy(a, "A");
d) a = 'A';

1.9 Wie gibt man den Wert einer double Variablen dv auf dem Bildschirm aus?
a) printf("%d",&dv);
b) printf("%lf",dv);
c) printf("%ld",(double)dv);
d) printf("%s",&dv);

1.10 Gegeben sei eine vierstellige Binärzahl 1011. Welchen Wert stellt diese Zahl im Dezimalsystem dar, wenn es sich um eine **unsigned** Zahl handelt?
a) 21 (10+11)
b) -4
c) 11
d) 3

1.11 Gegeben sei eine vierstellige Binärzahl 1011. Welchen Wert stellt diese Zahl im Dezimalsystem dar, wenn es sich um eine **signed** Zahl (Zweierkomplement) handelt?
a) -5
b) -11
c) -10
d) 8

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.

1.13 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)

1.14 Speicher der mit malloc angefordert wird befindet sich im folgenden Speicherbereich:
a) Stack (Stack Segment)
b) Heap (Heap Segment)
c) Globale Variablen (Daten Segment)
d) Programm Code (Code Segment)

1.15 Modulglobale Variablen befinden sich im folgenden Speicherbereich:
a) Stack (Stack Segment)
b) Heap (Heap Segment)
c) Globale Variablen (Daten Segment)
d) Programm Code (Code Segment)

1.16 Mit welchem Diagramm wird die Gesamtarchitektur eines C-Programmes beschrieben:
a) Structure Chart
b) Struktogramm
c) Flussdiagramm
d) Zeigerdiagramm

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

1.18 Welches ist die korrekte Definition eines Funktionszeigers, der auf die Funktion atof zeigen soll?
a) double (*fp)(char*);
b) double *fp(void);
c) void (*fp)(void);
d) void *fp (char*);

1.19 Arrays werden in C immer
a) call by value übergeben
b) rekursiv übergeben
c) als Zeiger auf Zeiger übergeben
d) call by reference übergeben

1.20 Welche Bedeutung kommt dem Zeichen 0x00 in C zu?
a) Es dient als Divisor
b) Es dient als Defaultwert bei allen Zeichen in einer lokalen Zeichenkette
c) Es wird oft als Übergabewert verwendet, wenn ein enum erwartet wird.
d) Es kennzeichnet das Ende einer Zeichenkette (EOS: End of String)

Aufgabe 2: **10 Punkte**

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

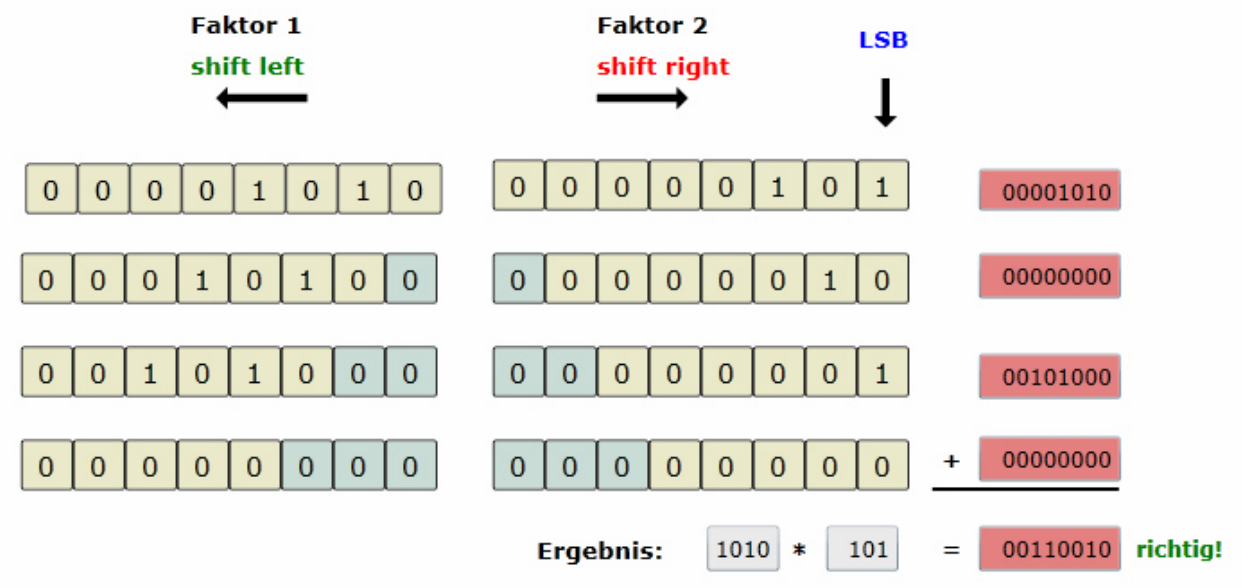
$1101\ 0101\ 0001_2 =$ 6521_8

$1101\ 0101\ 0001_2 =$ $D51_{16}$

$12_{13} =$ 15_{10}

$17_{10} =$ 11_{16}

2.2 Führen Sie eine Binär-Multiplikation 1010 * 101 durch. Die Abbildung ist aus dem Lernmodul entnommen, welches Sie im Labor verwendet haben. Komplettieren Sie die Abbildung. Der Rechenweg muss durch den vorgegebenen Algorithmus ersichtlich sein. (6 Punkte)



Jeweils 1 P für jede Zeile und 2 P für das Ergebnis

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 Label
{
   int iGermany;
   int iFrance;
};
struct Motor
{
   union Label uLabel;
   char* pcName;
   int   iPower[2];
   float fcostsinternal;
};
#define SIZE 10
```

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

mymotor.uLabel.iFrance = 1000;
mymotor.uLabel.iGermany = 2222;
mymotor.pcName = (char*) malloc (SIZE*2*sizeof(char));
strcpy(mymotor.pcName,"PowerMot-4711");
mymotor.iPower[0]=10;
mymotor.iPower[1]=20;
mymotor.fcostsinternal=520.5f;

printf("%d\n",sizeof(char));           // 1__________ 1 P
printf("%d\n",sizeof(char*));          // 4__________ 1 P
printf("%d\n",sizeof(esem1));          // 4__________ 1 P
printf("%d\n",sizeof(union Label));    // 4__________ 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)); // 13_________ 1 P
printf("%d\n",sizeof(struct Motor*));  // 4__________ 1 P
printf("%d\n",(esem2-esem1));          // -3_________ 1 P
printf("%d\n",mymotor.uLabel.iFrance); // 2222_______ 1 P

printf("%s\n", strtok(mymotor.pcName,"-"));   // PowerMot_____ 2 P
printf("%d\n", (mymotor.pcName[3]-mymotor.pcName[1]));  // -10 _ 2 P
printf("%d\n", (&mymotor.pcName[3]-&mymotor.pcName[1]));// 2____ 2 P

uc1 = 0xF2;
uc2 = 0x0F;
printf("%x\n",(uc1 | (uc2>>2)));       // f3_______ 2 P
printf("%x\n",((uc1^uc2) & 0xff));     // fd_______ 2 P
}
```

Aufgabe 4: (10 Punkte)
Über die Kommandozeile werden einem Programm zwei Parameter übergeben. Der erste Parameter gibt den Realteil und der zweite Parameter den Imaginärteil einer komplexen Zahl an. Das Programm soll den Betrag und die Phase (im Bereich -180 Grad bis +180 Grad ausgeben) dieser komplexen Zahl ausgeben. Falls die Anzahl der übergebenen Parameter nicht stimmt, soll eine Fehlermeldung ausgegeben werden. Ist die Anzahl der übergebenen Parameter korrekt, so wird an das Betriebssystem eine 1 zurückgegeben, sonst eine 0. Programmieren Sie strukturiert (strukturierte Programmierung)!
double sqrt(double);
double atan2(double y, double x);

```
#include <string.h>
#include <stdio.h>
#define _USE_MATH_DEFINES   // fuer M_PI
#include <math.h>

int main(int argc, char* argv[]) // 2 P für richtige Parameter
{
double dreal, dimg, dbetrag, ddeg,dgrad;
int iret;

if (argc == 3)
{
  dreal = atof(argv[1]); // 1 P
  dimg = atof(argv[2]); // 1 P
  dbetrag = sqrt(dreal*dreal+dimg*dimg); // 1P
  dgrad = atan2(dimg,dreal); // 1 P
  ddeg = (dgrad/M_PI)*180.; //1 P
  printf("Betrag: %lf - Winkel: %lf Grad",dbetrag,ddeg); // 0.5 P
  iret = 1;
}
else
{
  printf("Ungueltige Anzahl Parameter"); // 0.5 P
  iret = 0;
}

return iret; // 2 P für Strukt. Programmierung 0,1 Rückgabe, if/else
}
```

Aufgabe 5: (15 Punkte)

5.1 Tragen Sie jeweils den ASCII-Code als Binärzahlen für die Zeichen 'A' und 'm' in die Tabelle ein (1 Punkt) und berechnen Sie was sich jeweils durch die bitweise Exclusiv-Oder (EXOR) Operationen ergibt (2 Punkte).

'A'	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
'm'	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
'A'^'m'	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**0**
'm'	**0**	**1**	**1**	**0**	**1**	**1**	**0**	**1**
('A'^'m') ^ 'm'	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**

Welche Erkenntnis schließen Sie daraus, wenn Sie die erste und letzte Zeile vergleichen? (2 Punkte)

Wird ein Zeichen zweimal nacheinander mit dem gleichen Schlüsselzeichen mit EXOR verschlüsselt, so erhält man wieder den Anfangswert des Zeichens

5.2 Schreiben Sie eine Verschlüsselungsfunktion (fr. chiffrement) mit dem Funktionsnamen **vf**, die eine Verschlüsselung realisiert. (10 Punkte)

Input:

- Zeichenkette, die verschlüsselt werden soll
- Ein Zeichen als Schlüssel

Rückgabe:

- Verschlüsselte Zeichenkette

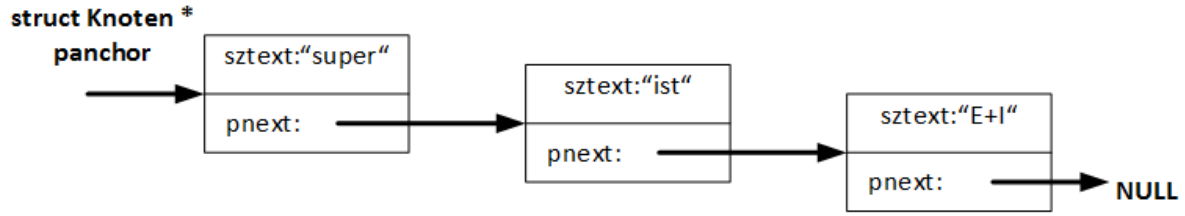
Bei der Verschlüsselung wird jedes Zeichen der übergebenen Zeichenkette mit dem Schlüssel bitweise Exklusiv-Oder (EXOR) verknüpft und die so verschlüsselte Zeichenkette wird als Rückgabewert zurückgegeben.

```
char* vf (char* ps, char key) // Parameter, Rückgabewert (1 + 1 P)
{
  char * pret = ps; // 1 P für das Merken

  while (*ps != 0x00) // 2 P
  {
    *ps = *ps^key; // 3 P
    ps++; // 1 P
  }

  return pret; // 1 P
}
```

Aufgabe 6: (25 Punkte)

Gegeben sei eine Struktur Knoten. Damit lassen sich verkettete Listen aufbauen, indem die Struktur Knoten einen Zeiger pnext auf eine Struktur Knoten enthält. Dieser Zeiger zeigt auf den nächsten Knoten oder enthält den Wert NULL, wenn das Ende der verketteten Liste erreicht ist.	`struct Knoten`
	`{`
	`char sztext[20];`
	`struct Knoten * pnext;`
	`};`

Die folgende Abbildung zeigt exemplarisch eine solche verkettete Liste. panchor ist dabei der Zeiger, der auf den ersten Knoten zeigt.



Implementieren Sie eine **rekursive** Funktion printallKnoten, welche einen Zeiger auf einen Knoten übergeben bekommt (z.B. panchor) und ausgehend von diesem Zeiger immer den sztext aller Knoten in umgekehrter Reihenfolge ausgibt, bis das Ende der Liste erreicht ist. Die Funktion soll zudem die Anzahl der Knoten in der Liste zurückgeben.
Wird die **rekursive** Funktion printallKnoten mit panchor für das obige Beispiel aufgerufen, so wird ausgegeben: „E+I ist super“ und der Rückgabewert ist 3.

```
unsigned int printallKnoten (struct Knoten * pk)
// Parameter, Rueckgabewert (1 + 1 P)
{
   unsigned int iret = 0; // Korrekte Berechnung Rückgabewert 5 P
   if (pk->pnext == NULL) // Abbruchkriterium 3 P
   {
     iret = 1;
   }
   else
   {
     iret = 1 + printallKnoten(pk->pnext); // Rekursiver Aufruf 10 P
   }

   printf("%s ",pk->szname); // Reihenfolge 5 P
   return iret;
}
```

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