Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Studiengang ESTB / EEEB

Note / normierte Punkte

Klausur in Programmieren

Sommer 2016, 21. Juli 2016

Dauer: 1,5h

Hilfsmittel: Keine (Wörterbücher sind auf Nachfrage erlaubt)

Name:
MatrikelNr.:

Aufgabe	1	2	3	4	5	6	Summe
Punkte max	12	12	21	14	26	15	100
Punkte							

Alle Fragen beziehen sich auf den Stoff der Vorlesung. Somit sind sie z.B. bezogen auf die Programmiersprache C. Auch sonst gelten die Konventionen wie in unserer Vorlesung.

1. Aufgabe: Grundlagen

Stellen Sie die nachfolgenden Zahlen in den jeweiligen Zahlensystemen dar (12P):

a) 1010 1011 1100 0111 (binär)	hex:	0xABC7	(1P)
	oktal:	0125707	(3P)
b) 231 (dezimal)	hex:	0xE7	(2P)
	binär:	1110 0111	(2P)
c) 7B (hex)	dezimal:	123	(3P)
	binär:	0111 1011	(1P)

2. Aufgabe: Grundlagen

Schreiben Sie ein vollständiges und lauffähiges C-Hauptprogramm, das zwei ganzzahlige Werte von der Konsole einliest, Minimum und Maximum ausgibt, sowie den Mittelwert aus beiden Werten bestimmt. Der Mittelwert soll in eine float-Variable gespeichert und dann ausgegeben werden. Berücksichtigen Sie bei der Ausgabe auf die Konsole eine passende Beschriftung (12 P)

```
#include <stdio.h>

int main()
{
  int iValue1;
  int iValue2;
  float fMittelwert;

  printf("Wert1: ");
  scanf_s(&iValue1);

  printf("Wert2: ");
  scanf_s(&iValue2);

  if(iValue1 > iValue2)
  {
    int iTmp = iValue1;
    iValue1 = iValue2;
    iValue2 = iTmp;
  }

  fMittelwert = (iValue1 + iValue2) / 2.0;

  printf("Minimum: %d\n", iValue1);
  printf("Maximum: %d\n", iValue2);
  printf("Mittelwert: %f\n", fMittelwert);

  return  0;
}
```

- Lösen Sie die Aufgaben bitte auf dem Blatt -

3. Aufgabe: Variablen und Schleifen

a) Erklären Sie die **wesentlichen** Unterschiede zwischen den folgenden Funktionsaufrufen in **wenigen** und **kurzen** Sätzen (bei Romanen kann es Punktabzug geben!): (12 P)

1. void functionCall1(int iInValue)
2. void functionCall3(int* piInOutValue)
3. int functionCall4()

1. call by value-Aufruf, nur der Wert wird als Kopie übergeben
2. call by reference-Aufruf, die Variable wird als Zeiger übergeben und kann geändert werden. Der Zeiger kann auch als Array interpretiert werden.
3. hier wird keine Variable übergeben, nur ein Funktionswert wird zurückgegeben

b) Welche Schleifenarten gibt es in C? Geben Sie kleine Beispiele als Programm**fragmente** (keine Funktionen, kein Hauptprogramm, keine Ein- oder Ausgabe!). (9 P)

```
int iPos=10;
while(iPos > 0)
{
  --iPos;
}

--------------------------------

int iPos=0;
do
{
  iPos++;
}
while(iPos < 10);

--------------------------------

int li;
for(li=0; li<20; li++)
{
}
```

- Lösen Sie die Aufgaben bitte auf dem Blatt -

4. Aufgabe: Array/Feld, Indizierung

a) Erklären Sie den Unterschied zwischen einem Array und einer Struktur **(kein Roman – kann Punktabzug geben!)**. (4 P)

Bei einem Array besitzen alle Elemente des Arrays denselben Datentyp, bei einer Struktur können sie unterschiedliche Datentypen besitzen. Der Zugriff bei Feldelementen eines Arrays erfolgt über einen Index, bei einer Struktur erfolgt der Zugriff auf die Strukturelemente über den Punktoperator.

b) Schreiben Sie eine kleine Funktion, bei der ein Array mit reellen Messwerten übergeben wird und das Minimum bestimmt wird. Das Minimum soll als reelle Zahl in Form eines Funktionswertes zurück gegeben werden. Geben Sie anschließend ein Programmfragment an, bei dem die von Ihnen definierte Funktion aufgerufen wird (**d.h. weniger als 5 Anweisungen reichen und bitte kein Hauptprogramm oder irgendwelche Ein- oder Ausgaben schreiben!). Bibliotheken dürfen nicht verwendet werden! Kann Abzug geben!** (10 P)

```
float minimum(float* afInValues, int iInCount)
{
  float fMinimum = afInValues[0];
  int li;

  for(li=1; li < iInCount; ++li)
  {
    if(fMinimum > afInValues[li])
    {
      fMinimum = afInValues[li];
    }
  }
  return fMinimum;
}

  float afValues[3];
  float fMinimum;
  afValues[0] = 50.5;

  fMinimum = minimum(afValues, 3);
```

- Lösen Sie die Aufgaben bitte auf dem Blatt -

5. Aufgabe: Zeichenketten

a) Schreiben Sie eine Funktion strlength, die die Länge einer mit 0 terminierten Zeichenkette bestimmt. Die Zeichenkette soll als Parameter übergeben werden. Der Funktionswert soll die Länge zurückgeben (kein Hauptprogramm, keine Ein- oder Ausgabe!). (10 P)

```
int strlength(char* acInString)
{
  int iIndex = 0;
  while(acInString[iIndex] != 0)
  {
    iIndex++;
  }
  return iIndex;
}
```

b) Schreiben Sie eine Funktion count, die die Anzahl von Großbuchstaben in einer Zeichenkette acString zählt. (kein Hauptprogramm, keine Ein- oder Ausgabe!). (16 P)

count("This IS An ExamPle!") → 6
Hinweis: Denken Sie daran, dass die Zeichen 'A' … 'Z' wie Zahlen benutzt werden können.

```
unsigned int countUpper(char* acInOutString)
{
  int iIndex = 0;
  unsigned int uiCount = 0;

  while(acInOutString[iIndex] != 0)
  {
    char cCurrent = acInOutString[iIndex];
    if('A' <= cCurrent  &&  cCurrent <= 'Z')
    {
      ++uiCount;
    }
    ++iIndex;
  }

  return uiCount;
}
```

- Lösen Sie die Aufgaben bitte auf dem Blatt -

6. Aufgabe: Algorithmus

Was macht die nachfolgende Funktion unknown_init? Wann ist das Ergebnis undefiniert und kann sogar einen Programmabsturz bewirken?

Bitte beschreiben Sie die Funktionsweise möglichst <u>abstrakt – Romane geben Abzug!</u> (15 P)

Hinweis: Testen Sie den Algorithmus anhand eines Funktionsaufrufs und beobachten Sie die Variablenwerte. Erinnern Sie sich an diese spezielle Form einer Funktion, diesmal allerdings ohne Ausgabe des Ergebnisses, dafür jedoch mit dessen Rückgabe.

```
#include <stdlib.h>

#define true 1
#define false 0
typedef int boolean;                    // moegliche Werte fuer boolean: true / false

typedef struct tStructureDef
{
  unsigned int   muiValue1;
  int*           mpValue2;
} tStructure;

tStructure initStructure(unsigned int uiInAnzahl, int iInDefault)
{
  tStructure  s;
  unsigned int i;
  s.muiValue1 = uiInAnzahl;
  s.mpValue2 = (int*) malloc(sizeof(int) *  s.muiValue1);
  for(i=0; i < s.muiValue1; ++i)
  {
    s.mpValue2[i] = iInDefault;
  }
  return s;
}

boolean unknown_init(unsigned int uiInAnzahl, int iInDefault, tStructure sInExpResult)
{
  unsigned int i;
  tStructure s = initStructure(uiInAnzahl, iInDefault);
  boolean value = (s.muiValue1 == sInExpResult.muiValue1? true : false);
  if(value)
  {
    for(i=0; i < s.muiValue1; ++i)
    {
      if(s.mpValue2[i] != sInExpResult.mpValue2[i])
        return false;
    }
  }
  return value;
}
```

unknown_init testet die Funktion initStructure (testcase_initStructure). sInExpResult ist dabei das erwartete Ergebnis.

Ein Programmabsturz ist möglich, wenn sInExpResult.mpValue2 auf kein passendes Array zeigt.