Dipl.Phys. Gerald Kempfer

Berliner Hochschule für Technik
- University of Applied Sciences -
Fachbereich VI – Informatik und Medien
Studiengang Technische Informatik Bachelor

Beispiel-Klausur zur Lehrveranstaltung Algorithmen und Datenstrukturen (TI-B) Sommersemester 2023

Hinweise:

Die Teilnahme an der Klausur ist nur bei Bestehen der Übungsaufgaben zulässig!

Zum Bestehen der Klausur benötigen Sie 50 Punkte von 100 möglichen Punkten.

Die Bearbeitungszeit der Klausur beginnt pünktlich um 16:00 Uhr und endet – ebenfalls pünktlich – um 17:30 Uhr. Sie haben also 90 Minuten Zeit.

Folgende Hilfsmittel in Papierform sind zugelassen: Skripte, Mitschriften und Bücher. **Nicht** zugelassen sind elektronische Geräte (Handys, Notebooks, PDAs, usw.).

Schreiben Sie mit Kugelschreiber oder Füller, **nicht** mit Bleistift! Verwenden Sie **nicht** die Farbe Rot. Schreiben Sie leserlich – was ich nicht lesen kann, ist grundsätzlich falsch! Beschreiben Sie nur die Vorderseiten!

Jeder Austausch mit anderen Personen wird als Täuschungsversuch gewertet und führt dazu, dass die Klausuren aller Beteiligten als „nicht bestanden“ gewertet werden.

Erläuterungen sollten kurz, aber dennoch präzise und vollständig sein. Wenn möglich ist eine stichpunktartige Beantwortung zu wählen, sofern die Verständlichkeit gegeben ist. Im Zweifelsfall können ganze Sätze Klarheit schaffen.

Kennzeichnen Sie jedes Blatt, das Sie abgeben, links oben mit Ihrer Matrikelnummer.

Viel Erfolg!

Name: _____

Matrikelnummer: _____ letzter Prüfungsversuch: ☐

Bewertung:

Aufgabe	Mögliche Punktzahl	Erreichte Punktzahl
1	10	
2	15	
3	25	
4	20	
5	30	
Summe	**100**	

Note Klausur _____

Aufgabe 1: Finden Sie im folgenden Programm alle Syntax- und Linkerfehler. Markieren Sie die Fehler und schreiben Sie hinter bzw. unter die Zeile, wie die Zeile richtig aussehen muss. (___ / 10)

```
01 #include <stdio.h>
02 #inklude <stdlib.h>
03
04 #define DEBUG
05
06 int main()
07 {
08    FEIL *Datei1, *Datei2;
09    char Quelle[] = 'datei1.txt";
10    char *Modus = "w";
11    int Zeichen1, Zeichen2;
12
13    Datei1 = fopen(Quelle, "r");
14    Datei2 = fopen("datei2.txt," Modus);
15
16    if (Datei1 && Datei2)
17    {
18       while (! feof(Datei4))
19       {
20          Zeichen1 := fgetc(Datei1);
21          if (Zeichen1 >= 0)
22          {
23             Zeichen2 = Zeichen1 + 17;
24             Zeichen2 %= 256++;
25             fputc(Zeichen2, Datei2);
26             #ifdef DEBUG
27                print("%c -> %c\n", Zeichen1, Zeichen2);
28             #endif
29          endif
30       }
31    }
32    #ifndef DEBUG
33    else
34    {
35       if (!Datei1)
36          printf("%s konnte nicht geoeffnet werden!", Quelle);
37       if (!Datei2)
38          printf("datei2.txt konnte nicht geoeffnet werden!");
39    }
40    #endif
41
42    fcloss(Datei1);
43    fclose(Datei2);
44
45    return 0;
46 }
```

Aufgabe 2: Multiple-Choice-Fragen. (___ / 15)
Kreuzen Sie an, wie der Satz richtig heißen muss. Es gibt immer nur eine richtige Antwort.

Die Definition einer Struktur wird mit folgendem Schlüsselwort eingeleitet:
- ❑ `define structure`
- ❑ `structure`
- ❑ `struct`

Ein Zeiger (mit dem Namen `Z`) auf eine Zeichenkette (mit einer Textlänge von max. 10 Zeichen) wird wie folgt definiert:
- ❑ `char Z[11];`
- ❑ `char * * Z[11];`
- ❑ `char * Z[11];`

Eine Zeiger-Variable auf einen Datenstrom hat in C den Datentyp
- ❑ STREAM *.
- ❑ FILESTREAM *.
- ❑ FILE *.

Der Präprozessor-Befehl `#include <stdio.h>` fügt die Headerdatei `stdio.h` ein aus dem
- ❑ Include-Verzeichnis.
- ❑ Windows-Systemverzeichnis.
- ❑ aktuellen Verzeichnis.

Mit `int **Zeiger;` wird
- ❑ beim Compilieren ein Fehler auftreten.
- ❑ ein Zeiger auf `int` definiert.
- ❑ ein Zeiger auf Zeiger auf `int` definiert.

Der Name einer Funktion
- ❑ ist ein Zeiger auf eine Variable.
- ❑ ist eine Variable.
- ❑ ist ein Zeiger auf die Funktion.

Mit `int const * Zeiger;` wird folgendes definiert:
- ❑ Ein veränderbarer Zeiger auf eine unveränderbare `int`-Variable.
- ❑ Ein unveränderbarer Zeiger auf eine unveränderbare `int`-Variable.
- ❑ Ein unveränderbarer Zeiger auf eine veränderbare `int`-Variable.

Die Anweisung `free(NULL);` bewirkt
- ❑ einen Speicherzugriffsfehler.
- ❑ eine Fehlermeldung.
- ❑ nichts.

Eine make-Datei ist
- ❑ eine Text-Datei.
- ❑ binär verschlüsselt.
- ❑ Passwort-geschützt.

Die `malloc`-Funktion ist deklariert in der Headerdatei
- ❑ `stdio.h`
- ❑ `stdlib.h`
- ❑ `string.h`

Um den binären Lesemodus beim Aufruf der `fopen`-Funktion anzugeben, muss folgender Modus angegeben werden:
- ❑ `"rb"`
- ❑ `"r" | "b"`
- ❑ `"r" + "b"`

Mit folgender Funktion aus der Headerdatei `stdlib.h` wird dynamisch Arbeitsspeicher reserviert:
- ❑ `memalloc`
- ❑ `malloc`
- ❑ `alloc`

Dateien mit langen Dateinamen (ab Windows 95) können nur geöffnet werden mit der Funktion
- ❑ `flfnopen` (*lfn* steht für *long file name*).
- ❑ `fopen` (wie alle anderen Dateien auch).
- ❑ können gar nicht geöffnet werden.

Mit `#error` wird gezielt eine
- ❑ Präprozessor-Fehlermeldung erzeugt.
- ❑ Compiler-Fehlermeldung erzeugt.
- ❑ Linker-Fehlermeldung erzeugt.

In einer `struct`-Variable
- ❑ können die verschiedenen Felder unterschiedliche Datentypen besitzen.
- ❑ müssen alle Felder den gleichen Datentypen besitzen.
- ❑ dürfen keine zwei Felder den gleichen Datentypen besitzen.

Aufgabe 3: Was gibt das folgende Programm aus? (___ / 25)
Schreiben Sie die Ergebnisse der vorgegebenen Ausdrücke sowie die Bildschirm-Ausgaben auf!

```
#include <stdio.h>

// Das folgende definiert "Funktionszeiger" als ein Zeiger auf
// eine Funktion, die ein char erhält und ein char zurückgibt:
typedef char (*Funktionszeiger)(char);

Funktionszeiger getFPtr(int);
void druckeZeichen(Funktionszeiger FPtr, char, int);

int main()
{
   Funktionszeiger FktPtr = NULL;

   druckeZeichen(getFPtr(1), 'A',  0);
   druckeZeichen(getFPtr(4), 'A', 17);
   druckeZeichen(getFPtr(2), 'b',  4);
   druckeZeichen(getFPtr(1), 'u', -1);
   druckeZeichen(getFPtr(0), 'f', -3);
   druckeZeichen(getFPtr(2), 'x', -3);
   druckeZeichen(getFPtr(3), 'x',  7);
   printf("\n");

   return 0;
}

char nextChar(char c)    {   return c + 1;   }
char prevChar(char c)    {   return c - 1;   }
char firstChar(char c)   {   return 'a';     }

Funktionszeiger getFPtr(int FktNr)
{
   switch (FktNr)
   {
      case 1: return nextChar;
      case 2: return prevChar;
      case 3: return firstChar;
   }
   return NULL;
}

void druckeZeichen(Funktionszeiger FPtr, char Zeichen, int Offset)
{
   if (FPtr)
      printf("%c", FPtr(Zeichen) + Offset);
}
```

Zwischenergebnisse: *(21 Punkte)*

`getFPtr(1)` zeigt auf Funktion _____

`FPtr('A')` = _____

`Fptr('A') + 0` = _____

`getFPtr(4)` zeigt auf Funktion _____

`FPtr('A')` = _____

`Fptr('A') + 17` = _____

`getFPtr(2)` zeigt auf Funktion _____

`FPtr('b')` = _____

`Fptr('b') + 4` = _____

`getFPtr(1)` zeigt auf Funktion _____

`FPtr('u')` = _____

`Fptr('u') + -1` = _____

`getFPtr(0)` zeigt auf Funktion _____

`FPtr('f')` = _____

`Fptr('f') + -3` = _____

`getFPtr(2)` zeigt auf Funktion _____

`FPtr('x')` = _____

`Fptr('x') + -3` = _____

`getFPtr(3)` zeigt auf Funktion _____

`FPtr('x')` = _____

`Fptr('x') + 7` = _____

Hinweis: Die Zeilen `FPtr(`*`Zeichen`*`)` und `FPtr(`*`Zeichen`*`) + `*`Offset` nur dann ausfüllen , wenn der Funktionszeiger kein `NULL`-Zeiger ist!

Bildschirm-Ausgabe des oben stehenden Programms: *(4 Punkte)*
(Beachten Sie dabei auch die Formatierungsangaben!)

Aufgabe 4: Schreiben Sie die drei fehlenden Funktionen zum vorgegebenen Hauptprogramm.
Die Funktion `addPlayer` soll den übergebenen Spieler in der verketteten Liste am Ende anhängen. Als Parameter wird ein Zeiger auf den Speicherbereich des Spielers (`TPlayer`) übergeben. Zurückgegeben wird nichts.
Die Funktion `printTeam` soll eine Liste von allen Spielern einer Mannschaft auf dem Bildschirm ausgeben (siehe Beispielausgabe). Als Parameter wird die gewünschte Mannschaft als Zeichenkette übergeben.
Die Funktion `deleteAllPlayer` soll die reservierten Speicherbereiche der Spieler wieder freigeben und die Spielereinträge aus der verketteten Liste wieder entfernen. Parameter und Ergebnis sind void. (___ / 20)

Beispielausgabe:

```
Mannschaftsliste Deutschland:
01: Thomas Mueller (Tore 6)
02: Lukas Podolski (Tore 2)

Mannschaftsliste Niederlande:
01: Arjen Robben (Tore 2)
02: Wesley Sneijder (Tore 5)

Torschuetzenkoenig: Thomas Mueller mit 6 Toren
```

Vorgegebenes Programm:

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

// Datentyp Fussballspieler:
typedef struct FS
{  char *Name;
   int Goals;
   char *Teamname;
   struct FS *Next;
} TPlayer;

TPlayer *First = NULL;
TPlayer *Last  = NULL;

TPlayer *createPlayer(char *, int, char *);
void printTopscorer();

// Platz für Ihre Funktionsdeklarationen (3 Punkte):




int main()
{  addPlayer( createPlayer("Frank Ribery",      3, "Frankreich" ) );
   addPlayer( createPlayer("Lionel Messi",      1, "Argentinien") );
   addPlayer( createPlayer("Thomas Mueller",    6, "Deutschland") );
   addPlayer( createPlayer("Lukas Podolski",    2, "Deutschland") );
   addPlayer( createPlayer("Arjen Robben",      2, "Niederlande") );
   addPlayer( createPlayer("Cristiano Ronaldo", 1, "Portugal"   ) );
   addPlayer( createPlayer("Wesley Sneijder",   5, "Niederlande") );
   addPlayer( createPlayer("David Villa",       5, "Spanien"    ) );

   printTeam("Deutschland");
   printTeam("Niederlande");

   printTopscorer();

   deleteAllPlayers();

   return 0;
}
```

```
/*****************************************************************************
 * Erzeugt einen neuen Spieler. Dazu wird für den Spieler Speicher reserviert
 * und die Daten des Spielers (Parameter) in diesen Speicher kopiert
 * Parameter: Name     - Name des Spielers
 *            Goals    - Anzahl der geschossenen Tore
 *            Teamname - Name der Mannschaft
 * Ergebnis:  Zeiger auf den reservierten Speicherbereich, in dem die Daten des
 *            Spielers hineinkopiert wurden; NULL-Zeiger, wenn kein Speicher
 *            reserviert werden konnte.
 *****************************************************************************/
TPlayer *createPlayer(char *Name, int Goals, char *Teamname)
{  TPlayer *Neu = malloc(sizeof(TPlayer));

   if (Neu)
   {  Neu->Name = malloc(strlen(Name) + 1);
      if (Neu->Name)
         strcpy(Neu->Name, Name);
      Neu->Goals = Goals;
      Neu->Teamname = malloc(strlen(Teamname) + 1);
      if (Neu->Teamname)
         strcpy(Neu->Teamname, Teamname);
   }

   return Neu;
}

/*****************************************************************************
 * Ermittelt den Torschützenkönig und gibt diesen auf dem Bildschirm aus. Sind
 * mehrere Spieler Torschützenkönig, wird nur der erste Spieler ausgegeben.
 * Parameter: keine
 * Ergebnis:  keins
 *****************************************************************************/
void printTopscorer()
{  TPlayer *temp = First;
   TPlayer *Topscorer = NULL;
   int MaxGoals = 0;

   while (temp != NULL)
   {  if (temp->Goals > MaxGoals)
      {  MaxGoals  = temp->Goals;
         Topscorer = temp;
      }
      temp = temp->Next;
   }
   if (Topscorer != NULL)
   {  printf("Torschuetzenkoenig: %s ", Topscorer->Name);
      printf("mit %i Toren\n\n", Topscorer->Goals);
   }
   else
      printf("Keinen Torschuetzenkoenig gefunden!\n\n");
}

// Platz für Ihre Funktionsdefinitionen (17 Punkte):
```

```
// Fortsetzung der Funktionsdefinitionen für Aufgabe 4:
```

Aufgabe 5: Schreiben Sie die drei fehlenden Funktionen zum vorgegebenen Hauptprogramm.
Konstante `MAX`: maximale Anzahl von Zeichenketten in der Datei
Konstante `MAXLEN`: maximale Länge einer Zeile in der Datei
Die Funktion `liesDatei` soll die Textdatei mit dem angegebenen Dateinamen einlesen und in dem angegebenen Array von Zeichenketten (übergeben wird die Adresse vom Zeiger auf das Array von Zeichenketten) ablegen. Der Speicherbereich für das Array sowie für die einzelnen Zeichenketten muss zuvor noch reserviert werden. Es kann eine korrekt-formatierte Textdatei vorausgesetzt werden!
Die Funktion `sortiere` soll das angegebene Array von Zeichenketten mittels der BubbleSort- und der Vergleichsfunktion sortieren.
Schließlich muss noch die Vergleichsfunktion erstellt werden, die zwei Zeichenketten erhält und diese miteinander vergleicht (z.B. mit `strcmp`). (___ / 30)

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

#define MAX 10
#define MAXLEN 100

typedef char * String;

void BubbleSort(String *Array, int Anzahl, int (*Vergleich)(String, String))
{  int i, j;
   char temp[MAXLEN];

   for (i = 1; i < Anzahl; i++)
      for (j = Anzahl - 1; j >= i; j--)
         if (Vergleich(*(Array + j), *(Array + j - 1)) < 0)
         {  strcpy(temp, *(Array + j));
            strcpy(*(Array + j), *(Array + j - 1));
            strcpy(*(Array + j - 1), temp);
         }
}

void schreibeDatei(String Name, String *AZ)
{  FILE *D;
   int i;

   D = fopen(Name, "w");
   if (D != NULL)
      for (i = 0; i < MAX; i++)
         fprintf(D, "%s\n", *(AZ + i));
   free(AZ);
}

// Platz für Ihre Funktionsdeklarationen (2 Punkte je Deklaration = 6 Punkte):




int main()
{  String *ArrayZeiger = NULL;

   liesDatei("unsortiert_char.txt", &ArrayZeiger);
   if (ArrayZeiger != NULL)
   {  sortiere(ArrayZeiger);
      schreibeDatei("sortiert_char.txt", ArrayZeiger);
   }
}
```

```
// Funktionsdefinitionen für Aufgabe 5 (24 Punkte):
```