**Technische Universität Hamburg-Harburg**
**Institut für Zuverlässiges Rechnen**
**Prof. Dr. S.M. Rump**

**WiSe 2018/19**

# Midterm Musterklausur Prozedurale Programmierung

*20. Dezember 2018*

Sie haben 90 Minuten Zeit zum Bearbeiten der Klausur.

Tragen Sie bitte zunächst Ihren Namen, Ihren Vornamen, Ihre Matrikelnummer, Ihr Studienfach und Ihr Studiensemester in **DRUCKSCHRIFT** in die folgenden jeweils dafür vorgesehenen Felder ein.

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Name: | | | | | | | | | | | | | | |
| Vorname: | | | | | | | | | | | | | | |
| Matr.-Nr.: | | | | | | | | Fach | | | | Sem. | | |

Es sind alle Aufgaben zu bearbeiten. Die maximal erreichbare Gesamtpunktzahl beträgt 60 Punkte. Zum Bestehen der Klausur sind 30 Punkte erforderlich.

| Aufgabe | Punkte | Korrektur |
|---|---|---|
| **1** | | |
| **2** | | |
| **3** | | |
| **4** | | |
| **5** | | |
| $\sum$ | | |

| | |
|---|---|
| Bonus | |
| **Note** | |

**Zur Beachtung:**
Schreiben Sie bitte weder mit Bleistift noch mit Rotstift. Es sind keinerlei Hilfsmittel in dieser Klausur zugelassen! Das Schreiben vor dem Startsignal und auch das Schreiben nach dem Endsignal führt ohne weitere Warnung sofort zur Ungültigkeit der Klausur. Dies gilt auch für das Schreiben von Namen und Matrikelnummer nach dem Endsignal. Vergessen Sie nicht, den „Vorbehalt“ zu unterschreiben.
**Programmieren Sie die Aufgaben in ISO-C oder ANSI-C.**

**Vorbehalt**
Ich bin darüber belehrt worden, dass die von mir zu erbringende Prüfungsleistung nur dann bewertet wird, wenn die Nachprüfung durch das Zentrale Prüfungsamt der TUHH meine offizielle Zulassung vor Beginn der Prüfung ergibt.

_____

(Datum, Unterschrift)

# Aufgabe 1 (12 Punkte)

Beantworten Sie die jeweiligen Fragen zu den Codezeilen.

**Hinweise:**

- Eine vollständig korrekt beantwortete Frage wird mit +1 Punkt, eine falsch beantwortete Frage mit −1 Punkt und eine unbeantwortete Frage mit 0 Punkten bewertet.
- Uneindeutige Antworten, insbesondere uneindeutige Korrekturen gelten als falsche Antwort.
- Die minimal mögliche Punktezahl ist 0 Punkte.

☐ Ja ☑ Nein

1. Öffnet die folgende Anweisung eine Datei zum Lesen?

```
FILE* fp = fopen("input.txt", 'r');
```

☑ Ja ☐ Nein

2. Lautet die `printf`-Ausgabe „llo“?

```
char *start = "hello";
char *end = start + 5;
printf("%s", ((end - start) / 2 ) + start);
```

☐ Ja ☑ Nein

3. Wurde für die String-Kopie genügend Speicher bereitgestellt?

```
char string[5];
strcpy(string, "hello");
```

☑ Ja ☐ Nein

4. Lautet die `printf`-Ausgabe „-1“?

```
unsigned i = -1;
printf("%d",i);
```

☐ Ja ☑ Nein

5. Ist folgende **`while`**-Schleife eine Endlosschleife?

```
int i = 1;
while(i!=-1) i++;
```

☐ Ja ☑ Nein

6. Ist `substr` ein Zeiger auf die null-terminierte Zeichenkette „bahn“?

```
char str[] = "autobahnhof";
char* substr = str + 4;
```

7. Gibt folgendes Programm das letzte Kommandozeilenargument aus?

☐ Ja ☑ Nein

```
#include <stdio.h>
int main (int argc, char* argv[]) {
  printf("%s", argc[argv-1]);
  return 0;
}
```

8. Eignet sich folgende Struktur für die Erstellung einer doppelt verketteten Liste?

☐ Ja ☑ Nein

```
struct Position{
  int x,y;
  struct Position next, prev;
};
```

9. Es sei A der Name eines zweidimensionalen Arrays. Sind die folgenden beiden Elementzugriffe gleichwertig?

☑ Ja ☐ Nein

```
A[4][5];
*(*(A+4)+5);
```

10. Bestimmt folgendes Makro das Maximum zweier Zahlen?

☐ Ja ☑ Nein

```
#define MAX(a, b) ((a) < (b) ? (a) : (b))
```

11. Lautet die Ausgabe „20“?

☐ Ja ☑ Nein

```
int a = 0;
for (int i = 0; i < 10; i++)
  a++;
  ++a;
printf("%d", a);
```

12. Bestimmt folgende rekursive Funktion die Anzahl an Ziffern einer Ganzzahl?

☑ Ja ☐ Nein

```
unsigned count(unsigned number){
  return (number >=  10 ? 1 + count(number/10) : 1);
}
```

# Aufgabe 2 (12 Punkte)

Sie haben die freie Auswahl in einem Elektro-Kaufhaus und dürfen Ihren Rucksack füllen. Ihre Lieblings-Artikel liegen in einer einfach verketteten Liste vor. Beachten Sie, dass der Vorrat (`stock`) der Artikel und das Maximal-Gewicht Ihres Rucksacks begrenzt sind.

a) Schreiben Sie eine rekursive Funktion, die zu einem vorgegebenen Maximal-Gewicht jede mögliche Artikel-Auswahl mit einem Gesamtgewicht (`weight`) kleiner oder gleich dem Maximal-Gewicht prüft und den maximalen Gesamtwert (`cost`) zurückgibt.

b) Schreiben Sie eine `main`-Funktion, welche Ihre rekursive Funktion mit dem Listenanfang und einem Maximal-Gewicht von 4500 aufruft und das Ergebnis ausgibt.

```c
#include <stdio.h>
struct item {
    int weight; // Artikelgewicht in Gramm
    int value;  // Artikelwert in Euro
    int stock;  // Vorhandene Artikelmenge
    struct item *next;
};
struct item laptop = {1500, 1000, 3, NULL};
struct item tablet = { 700,  300, 2, &laptop};
struct item mobile = { 180,  500, 2, &tablet};
struct item *items = &mobile;

// Nicht gefordert
int max(int a, int b){
    return a > b ? a : b;
}
// a)
int rucksack(struct item* item, int limit) {
    if(item == NULL) return 0;
    int sum = 0;
    for(int i = 0; i <= item->stock && i*item->weight <= limit; i++)
        sum = max(sum, i*item->value + rucksack(item->next, limit - i*item->weight));
    return sum;
}

// b)
int main()
{
    printf("Der maximale Wert ist: %d", rucksack(items, 4500));
    return 0;
}
```

# Aufgabe 3 (12 Punkte)

a) Schreiben Sie eine Funktion `quadrat`, die zu einer gegebenen Kantenlänge den Umfang und den Flächeninhalt eines Quadrats berechnet und an die aufrufende Funktion zurückgibt. Schreiben Sie eine `main`-Funktion, die Ihre Funktion mit geeigneten Parametern aufruft.

```
1  #include <stdio.h>
2
3  void quadrat(double l, double* U, double* F) {
4    *U = 4*l;
5    *F = l*l;
6  }
7
8  int main() {
9    double l = 3.0, U, F;
10   quadrat(l, &U, &F);
11   printf("Ein Quadrat mit Kantenlaenge l=%f ", l);
12   printf("hat den Umfang U=%f und den Flaecheninhalt F=%f\n", U, F);
13   return 0;
14 }
```

b) Welche zwei Möglichkeiten der Parameterübergabe haben Sie? Erklären Sie anhand der Funktion in a) beide Möglichkeiten!

1. Call-by-value: eine Kopie des Wertes der Variable wird übergeben. Der Wert in der aufrufenden Funktion bleibt unverändert. Die Gültigkeit der Kopie endet mit dem Funktionsrücksprung. Angewendet in der Funktion `quadrat` für die Variable l.
2. Call-by-Reference: eine Kopie der Adresse der Variable in der aufrufenden Funktion wird übergeben. Der Wert der Variable kann in der aufrufenden Funktion verändert werden. Auch mehrere Rückgabewerte können so in C realisiert werden, wie in der Funktion `quadrat` für die Parameter `U` und `F`.

c) Ein **int**-Array `a` soll mithilfe der Funktion `print_reverse` in umgekehrter Reihenfolge ausgeben werden. Diese Funktion soll indirekt über eine zweite Funktion `print_array` ausgeführt werden. Implementieren Sie `print_array` mit Eingabeparametern für `a`, `len` und einem Funktionszeiger für `print_reverse`. Rufen Sie `print_array` aus der vorgegebenen `main`-Funktion korrekt auf.
**Hinweis:** Die geforderte Implementierung von `print_array` ist mit drei Codezeilen realisierbar!

```
1  #include <stdio.h>
2
3  void print_reverse(int* a, int n) {
4    for (int i = n-1; i >= 0; i--) {
5      printf("%d ", a[i]);
6    }
7  }
8
9  void print_array(int* a, int n, void (*print)(int*, int)) {
10   print(a, n);
11 }
12
13 int main() {
14   int a[] = { 1, 2, 3, 4, 5 };
15   int len = 5;
16   print_array(a, 5, print_reverse);
17   return 0;
18 }
```

# Aufgabe 4 (12 Punkte)

Schreiben Sie ein Programm, welches $M$ unsortierte Datenzeilen mit jeweils $N$ ganzen Zahlen, durch Leerzeichen getrennt, aus einer Datei einliest und sortiert auf dem Bildschirm ausgibt. Der Name der Eingabedatei, sowie die ganzen Zahlen $M$ und $N$ werden in genannter Reihenfolge als Kommandozeilenargumente übergeben. Eine Sortierfunktion mit der Signatur `void sort(int* start, int length)` ist bereits eingebunden und muss für die Verwendung in ihrem Programm **nicht** implementiert werden. Die eingelesenen Datenzeilen sollen in einem dynamisch allozierten Array gespeichert werden. Ihr Programm hat zu prüfen, ob ausreichend viele Kommandozeilenargumente übergeben wurden, ob die Eingabedatei auslesbar ist und ob die dynamischen Speicherallokationen erfolgreich waren. Im Fehlerfall darf auf Fehlermeldungen verzichtet werden.

**Beipsiel:** Inhalt einer Eingabedatei mit $M = 4$ unsortierten Datenzeilen mit jeweils $N = 3$ ganzen Zahlen.

```
7 4 1
8 2 5
3 9 6
5 2 6
```

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

void sort(int* start, int length);  // Bereits implementiert!

int main (int argc, char* argv[]) {
  if (argc < 4) {
    exit(EXIT_FAILURE);
  }
  int M = 0;
  int N = 0;
  sscanf(argv[2], "%d", &M);
  sscanf(argv[3], "%d", &N);

  int* A = (int*) malloc(N * sizeof(int));
  FILE* fp = fopen(argv[1], "r");
  if ((A == NULL) || (fp == NULL)) {
    exit(EXIT_FAILURE);
  }

  for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
      fscanf(fp, "%d", &A[j]);
    }
    sort(A, N);
    for (int j = 0; j < N; j++) {
      printf("%d ", A[j]);
    }
    printf("\n");
  }

  fclose(fp);
  free(A);

  return EXIT_SUCCESS;
}
```

# Aufgabe 5 (12 Punkte)

a) Definieren Sie für eine einfach verkettete Liste eine Struktur `Person` mit folgenden Komponenten:

- `ID`: Für das gesamte Programm eindeutige ganze Zahl
- `name`: String mit maximal 50 sichtbaren Zeichen

b) Schreiben Sie eine Funktion `insert_n`, welche dynamisch ein neues Listenelement der Struktur aus Aufgabenteil a) erzeugt. Die Komponenten `ID` und `name` des neuen Listenelements sollen mit entsprechenden Eingabeparametern initialisiert werden und in eine bestehende Liste mit dem Anfangszeiger `start` an der Position mit dem Index $n$ einfügt werden. Der ggf. neue Listenanfang soll von ihrer Funktion zurückgegeben werden. Sollte die Liste mit dem Anfangszeiger `start` weniger als $n+1$ Elemente besitzen, wird das neue Element an das Ende der Liste hinzugefügt. Berücksichtigen Sie auch den Fall, dass `start` auf eine leere Liste zeigt.

c) Gehen Sie davon aus, dass zu Ihrer Struktur aus Aufgabenteil a) zwei nach dem Feld `ID` aufsteigend sortierte Listen mit den Anfangszeigern `A` und `B` existieren. Jede `ID` kommt in beiden Listen höchstens einmal vor. Schreiben Sie eine Funktion `merge`, welche die jeweils sortierten Listen `A` und `B` in eine einzige aufsteigend sortierte Liste überführt und den Anfangszeiger zurückgibt.

**Hinweis:** Besonders einfach als rekursive Funktion.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  // zu a)
6  struct Person {
7    int ID;
8    char name[51];
9    struct Person* next;
10 };
11 typedef struct Person sP;
12
13 // zu b)
14 sP* insert_n(sP* start, unsigned int n, int ID, const char* name) {
15   sP* new = (sP*) malloc(sizeof(sP));
16   new->ID = ID;
17   strcpy(new->name, name);
18   if ((n == 0) || (start == NULL)) {
19     new->next = start;
20     return new;
21   }
22   sP* iter = start;
23   for (; (iter->next != NULL) && --n; iter = iter->next);
24   new->next = iter->next;
25   iter->next = new;
26   return start;
27 }
28
29 // zu c)
30 sP* merge(sP* A, sP* B) {
31   if (A == NULL) return B;
32   if (B == NULL) return A;
33   if (A->ID > B->ID) {  // swap
34     sP* tmp = A;
35     A = B;
36     B = tmp;
37   }
38   A->next = merge(A->next, B);
39   return A;
40 }
41
```

```
42 // Ab hier nicht Teil der Aufgabe
43
44 void print_list(sP* list) {
45   printf("\n--------------------\n");
46   for (; list != NULL; list = list->next) printf("%d ", list->ID);
47 }
48
49 int main() {
50   sP *A = NULL, *B = NULL;
51   A = insert_n(A, 0, 1, "Eins");
52   A = insert_n(A, 1, 6, "Sechs");
53   A = insert_n(A, 1, 3, "Drei");
54   A = insert_n(A, 2, 5, "Fuenf");
55   B = insert_n(B, 0, 2, "Zwei");
56   B = insert_n(B, 1, 4, "Vier");
57   B = insert_n(B, 10, 7, "Sieben");
58   print_list(A);
59   print_list(B);
60   A = merge(A, B);
61   print_list(A);
62   return 0;
63 }
```