Studiengänge: Mechatronik, Informatik, Flug- und Fahrzeuginformatik

Prüfung
Grundlagen der Programmierung 1

Prüfer: Glavina, Regensburger, Schmidt
Prüfungsdauer: 90 Minuten
Hilfsmittel: keine

Studiengang	Dozent	Matrikelnummer	Semester	Raum	Platz

Aufgabe	1	2	3	4	5	Σ	Note
Punkte							

Bitte beachten:
Tragen Sie Ihre persönlichen Angaben auf dieses Deckblatt ein.
Schreiben Sie Ihre Antworten direkt in die dafür vorgesehenen freien Stellen des Angabetextes.
Geben Sie alle Blätter wieder ab, auch wenn einzelne Seiten nicht beschrieben sein sollten.
Alle Blätter der Angabe müssen bei der Abgabe wieder richtig sortiert und geheftet sein!

Viel Erfolg!

Aufgabe 1: (Programmverständnis und Syntax, ca. 15%)

a) Geben Sie für die Funktionen f1 bis f5 jeweils den Funktionskopf (Prototyp) an. Die Typen lassen sich aus den Aufrufen der Funktionen in der Funktion `main` ermitteln.

```
#include <stdio.h>

struct element {
    char* name;
    struct element* next;
};

int main(void) {
    struct element* liste = NULL;
    int x = f1("xyz", 'x');
    char *s = f2(&x, liste);
    f3(&liste, s + 1);
    printf("%c", f4(s[2], *(s + 1)));
    printf("%s", f5(liste->next)->name);
    return 0;
}
```

****
f1:

****
f2:

****
f3:

****
f4:

****
f5:

b) Welche Ausgaben erzeugt das folgende Programm?

```c
#include <stdio.h>
#define N 5

void f(int a[], int n) {
    int k;
    for (k=0; k<n; k++) {
        if (k<2) {
            a[k] = 1;
        } else {
            a[k] = a[k-2] + a[k-1];
        }
    }
}

int main (void) {
    int i;
    int feld[N];

    f(feld,N);
    for (i=0; i< N; i++) {
        printf("feld[%d] = %d\n",i,feld[i]);
    }
    return 0;
}
```

Aufgabe 2: (Algorithmenentwurf, ca. 20 %)

Schreiben Sie eine Prozedur `drucke`, welche das Bild der Ziffer 5 aus Sternchen zusammengesetzt auf dem Bildschirm ausgibt. Die Ziffer ist gemäß einem Parameter `n` wie nachfolgend gezeigt zu skalieren:

n = 1

```
*****
*
****
    *
    *
****
```

n = 2

```
******
*
*
*****
     *
     *
     *
*****
```

n = 3

```
*******
*
*
*
******
      *
      *
      *
      *
******
```

n = 4 *usw.*

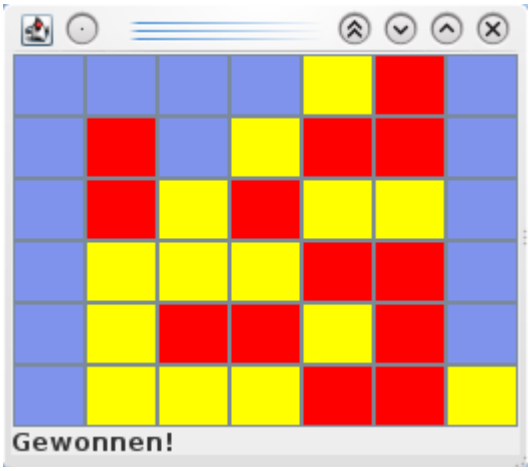
```
********
*
*
*
*
*******
      *
      *
      *
      *
      *
*******
```

Hinweis:
Falls Sie Hilfsfunktionen verwenden, die nicht in der C-Standardbibliothek enthalten sind, müssen diese hier implementiert werden.

Aufgabe 3: (Arrays, ca. 25 %)

Im Spiel "Vier gewinnt" setzen zwei Spieler (hier: rot und gelb) abwechselnd jeweils einen Stein. Gewonnen hat, wer als erster vier Steine seiner Farbe horizontal, vertikal oder diagonal anordnen kann. Im nachstehend gezeigten Beispiel hat gelb gewonnen, da vier gelbe Steine diagonal platziert werden konnten:

Hinweis bei Druck in Graustufen:
Das Feld rechts unten ist gelb und das Feld links oben ist blau.
Die dritte verbleibende Graustufe ist also rot.



Das Spielbrett werde in C durch ein globales Array dargestellt:

```
#define BLAU 0
#define GELB 1
#define ROT  2

#define ZEILEN 6
#define SPALTEN 7

int brett[ZEILEN][SPALTEN];
```

Das obere linke Feld des Bretts habe die Indizes (0, 0). Anfangs seien alle Felder mit der Farbe BLAU initialisiert (steht für ein nicht-belegtes Feld). Das Belegen mit einem Spielstein kann durch Zuweisung einer entsprechenden Farbe beschrieben werden:

```
brett[3][4] = ROT;
```

Weiter auf nächster Seite.

Schreiben Sie eine Funktion

```
int hat_gewonnen(int zeile, int spalte)
```

die prüft, ob der letzte Zug (dessen Indizes als Parameter übergeben werden) ein Gewinnzug war; wenn ja, wird 1 zurückgegeben, ansonsten 0.

Hinweis:
Falls Sie Hilfsfunktionen verwenden, die nicht in der C-Standardbibliothek enthalten sind, müssen diese hier implementiert werden.

Aufgabe 4: (Zeichenketten, ca. 15 %)

Gegeben sei folgendes Programm:

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

int main(void) {
    char *sub = substring("Hamburger", 3, 7);

    printf("%s\n", sub);
    free(sub);
    return 0;
}
```

Realisieren Sie die Funktion `substring`, die innerhalb eines Strings einen Teil-String ermittelt und zurückgibt. Die Funktion habe drei Parameter:

- der erste Parameter ist der String, aus dem der Teil-String zu entnehmen ist
- der zweite Parameter ist der Index `begin`
- der dritte Parameter ist der Index `end`

Der Teilstring beginnt bei `begin` und erstreckt sich bis `end` – 1; die Länge des Teilstrings ist also `end` – `begin`. Im gezeigten Beispiel ergibt sich als Teilstring `"burg"`.

Die Funktion soll in folgenden Fällen `NULL` zurückgeben:

- falls der an `substring` übergebene String `NULL` ist
- falls `begin` negativ ist
- falls `begin` größer als `end` ist
- falls `end` größer als die Länge des Strings ist

Der übergebene String darf nicht verändert werden.

// Fortsetzung auf nächster Seite möglich

Aufgabe 5: (NULL-terminierte Arrays, ca. 25 %)

Das nachstehende Programm arbeitet mit einem statisch vereinbarten Array, dessen Elemente Pointer auf Zeichenketten sind. Die Pointer zeigen auf Zeichenketten, die dynamisch alloziert werden.

In Analogie zu Zeichenketten wird vereinbart, dass der kleinste Index, dessen Eintrag ein NULL-Pointer ist, das aktuelle Ende des Arrays signalisiert. Alle Pointer-Einträge mit höherem Index werden ignoriert. Das Array muss zu jedem Zeitpunkt mindestens einen NULL-Pointer als Eintrag haben.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_NUM 10

int main (void) {
    char* helden[MAX_NUM + 1];

    helden[0] = NULL;
    add(helden, "Frodo Beutlin",  MAX_NUM+1);
    add(helden, "Luke Skywalker", MAX_NUM+1);
    add(helden, "Harry Potter",   MAX_NUM+1);
    add(helden, "Eragon",         MAX_NUM+1);
    drucke(helden);
    freigabe(helden);
    return 0;
}
```

Im gezeigten Beispiel werden einem anfangs leeren Array vier Elemente hinzugefügt, danach wird das Array auf dem Bildschirm ausgegeben. Die Ausgabe lautet:

```
Frodo Beutlin,Luke Skywalker,Harry Potter,Eragon
```

Als letztes wird der dynamisch allozierte Speicher wieder freigegeben.

a) Schreiben Sie die Prozedur `drucke`, welche alle Elemente vor dem aktuellen Ende des Arrays auf den Bildschirm ausgibt. Die Zeichenketten sollen dabei, wie oben gezeigt, durch Kommata getrennt werden.

b) Schreiben Sie die Prozedur `freigabe`, die den für die Zeichenketten dynamisch allozierten Speicher wieder freigibt.

c) Schreiben Sie die Prozedur `add`, welche ein neues Element am aktuellen Ende des Arrays einfügt. Da die Funktion wiederverwendbar sein soll, müssen Sie die Zeichenketten vor dem Einfügen duplizieren und den dafür notwendigen Speicher dynamisch anfordern. Aus dem gleichen Grund wird die maximale Größe des Arrays als Parameter mitgegeben.
Falls das Array voll ist, soll der Text `"Array voll"` ausgegeben werden. Falls kein Speicher mehr alloziert werden kann, soll `"Heap voll"` ausgegeben werden.

(Ende des Aufgabentextes) Viel Erfolg!