Übungsklausur
am 13.12.23 bei Prof. Dr.-Ing. Maike Stern

OTH Regensburg, Fakultät für Informatik und Mathematik
Studiengang Informatik, Programmieren 1 Wintersemester 2023 / 2024
Bearbeitungszeit: 90 Minuten

Hinweise:

- Diese Klausur besteht aus 12 Seiten (inklusive des Deckblatts) und 12 Aufgaben. Überprüfen Sie Ihre Klausur auf Vollständigkeit
- Als Hilfsmittel ist ein Taschenrechner zugelassen
- Verwenden Sie für die Lösungen den dafür vorgesehenen Platz direkt bei der jeweiligen Aufgabe
- Schreiben Sie leserlich
- Die Programmiersprache ist C
- Achten Sie beim Programmieren auf saubere Formatierung (Einrückungen etc.)
- Wenn Sie mehr Platz benötigen, verwenden Sie die Rückseite. Geben Sie in diesem Fall klar an, dass Sie die Rückseite verwendet haben.

Aufgabe	Punkte	Erreicht
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12 a		
12 b		
Total:		

Note: …….

Note, Unterschrift Erstprüfer: ……..

ggf. Unterschrift Zweitprüfer: ……

1. Welche der folgenden Aussagen sind richtig bzw. falsch? Kreuzen Sie entsprechend an. Jede richtige Antwort gibt 1 Punkt, jede falsche Antwort einen Minuspunkt. Mindestens 0 Punkte. (17 Punkte)

	Aussage	richtig	falsch
1	Kommentare dürfen in C nicht mehrzeilig sein		
2	In C kann man Funktionen sowohl aus Bibliotheken verwenden als auch selbst-definierte Funktionen		
3	In C ist es verboten, innerhalb von selbst definierten Funktionen weitere Funktionen aufzurufen		
4	Funktionsaufrufe besitzen immer ein Paar runde Klammern hinter dem Funktionsnamen		
5	Prototypen müssen hinsichtlich Rückgabetyp und Funktionsparameter mit der Definition der jeweiligen Funktion übereinstimmen		
6	Bei der Variablendeklaration machen Sie den Compiler mit einem Namen (Bezeichner) für die Variable bekannt und verknüpfen diesen Namen mit einem Typ		
7	Bei Funktionsnamen wird nicht auf Groß- bzw. Kleinschreibung geachtet		
8	Ein *String* ist ein Basisdatentyp in C		
9	Mit dem sizeof-Operator können Sie herausfinden, wieviel Speicherplatz eine Variable eines bestimmten Datentyps beansprucht		
10	Wird eine Variable nicht initialisiert, so gibt der Compiler einen Fehler aus		
11	%u ist der Formatspezifizierer einer Float-Variable		
12	Es ist in C nicht erlaubt, eine if-Bedingung ohne else-Verzweigung zu verwenden		
13	Bei einer do-while-Schleife wird der Anweisungsblock mindestens einmal durchlaufen		
14	Wird ein Anweisungsblock verlassen, so wird der Speicherplatz aller lokalen Variablen die in ihm definiert wurden freigegeben		
15	Sie können in C die Adresse einer Variablen mit Sternchen-Operator * ermitteln und diese in einer beliebigen Variablen speichern		
16	Die Adresse einer Variablen wird bei ihrer Erzeugung bestimmt und bleibt über die Lebensdauer konstant		
17	Der Name eines Arrays entspricht der Anfangsadresse des ersten Arrayelements		

2. Entwerfen Sie eine while-Schleife, die sich wie eine if-Bedingung verhält. Das heißt, die while-Schleife soll im Code die gleiche Wirkung haben, wie folgende if-Bedingung:
(4 Punkte)

```
1     If (<Bedingung>) {
2       <Anweisungen>;
3     }
4
```

Antwort:

3. Schreiben Sie eine for-Schleife, die die geraden Werte von 0 bis inklusive-20 ausgibt (also 0, -2,-4, …,-20).
(4 Punkte)

Antwort:

4. Wie wandeln Sie eine als String eingelesene bzw. gespeicherte Zahl in den repräsentierten Wert um? Also z.B. der String "5" in die Integerzahl 5.
(2 Punkte)

Antwort:

5. Betrachten Sie folgenden C-Code:

```
1     #include <stdio.h>
2
3     int main() {
4       int var_1 = 1, var_2 = 2, var_3 = 3;
5
6       var_1 += var_2;
7       var_3 -= var_1;
8
9       printf("var_1 = %i, var_2 = %i, var_3 = %i", var_1, \
10              var_2, var_3);
11
12      return 0;
```

Welche Werte werden durch das printf-Statement ausgegeben?
Tragen Sie die Werte in die Tabelle ein.
(3 Punkte)

Variable	Wert
var_1	
var_2	
var_3	

6. Bugs finden & fixen. Betrachten Sie den folgenden C-Code. Warum ist der Code fehlerhaft? Begründen Sie Ihre Antwort und nutzen Sie die Zeilennummern.
(5 Punkte)

```
1    #include <stdio.h>
2
3    typedef struct Person{
4      int age;
5      char firstName[30];
6      char lastName[30];
7    } Person
8
9    struct person *createPerson(char *firstN, char *lastN, int age){
10     Person person;
11     person.age = age;
12     strcpy(person.firstName, firstN);
13     strcpy(person.lastName, lastN);
14
15     return &person;
16   }
```

Antwort:

7. Betrachten Sie das folgende C-Programm. Welche Ausgaben erzeugt es?
(6 Punkte)

```c
#include <stdio.h>

int function_1(int input);
int function_2();

int i;

int main(){
  int j;
  i = 5, j = 2;

  printf("i ist %i, j ist %i\n", i, j);

  i = function_1(j);
  printf("i ist %i, j ist %i\n", i, j);

  i = function_2();
  printf("i ist %i, j ist %i\n", i, j);

  return 0;
}


int function_1(int input){
  i = i * 2;
  function_2();
  return input + 5;
}

int function_2(){
  int j;
  for (j = 0; j < 10; j++){
     i += 1;
  }
  return i;
}
```

Antwort:

8. Bugs finden & fixen. Was ist an dem folgenden Programm falsch und warum kann es sein, dass man den Fehler nicht sofort erkennt?
(5 Punkte)

```
1    #include <stdio.h>
2
3    int main(){
4      int number = 13;
5      int *pointer;
6      *pointer = number;
7
8      printf("number hat den Wert: %i\n", number);
9      *pointer = 12;
10     printf("number hat den Wert: %i\n", *pointer);
11
12     return 0;
13   }
14
```

Antwort:

9. Analysieren Sie das folgende C-Programm und geben Sie an, welche Ausgaben es erzeugt.

```c
#include <stdio.h>

void swap(int *first, int *second);
void swapSomeMore(int *array, int firstOffset, int secondOffset);
void printArray(int *array, int length);

int main(){
  int array[5] = {1, 2, 3, 4, 5};
  printArray(array, 5);

  swap(&array[2], &array[4]);
  printArray(array, 5);

  swapSomeMore(array, 0, 1);
  printArray(array, 5);
}

void swap(int *first, int *second){
  int temp;
  temp = *first;
  *first = *second;
  *second = temp;
}

void swapSomeMore(int *array, int firstOffset, int secondOffset){
  int temp;
  temp = *(array + firstOffset);
  array[firstOffset] = *(array + secondOffset);
  array[secondOffset] = temp;
}

void printArray(int *array, int length){
  for(int i = 0; i < length; i++){
     printf("%i ", array[i]);
  }
  printf("\n" );
}

```

Antwort:

10. Schreiben Sie eine rekursive Funktion, die zwei vorzeichenlose Integerwerte miteinander multipliziert, ohne den Multiplikationsoperator zu verwenden. Das Ergebnis wird als Rückgabewert geliefert.
Beispiel: 3*4 kann auch gerechnet werden als 3+3+3+3.
Globale Variablen dürfen nicht verwendet werden.
(6 Punkte)

Funktionssignatur:

```
unsigned int multiply(unsigned int a, unsigned int b)
```

Antwort:

11. Schreiben Sie eine Funktion, die die Länge eines Strings ohne das String-Ende-Zeichen zurückliefert. Verwenden Sie dafür keine Funktion aus der Standardbibliothek.
(4 Punkte)

Signatur der Funktion:

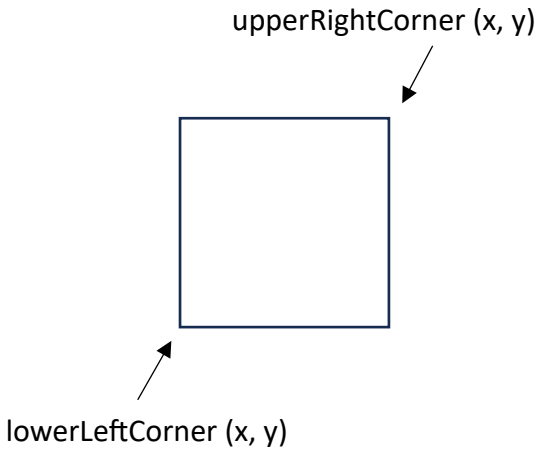
```
int myOwnStrLen(char *string)
```

Antwort:

12. Arbeiten mit Structs
(10 Punkte)

Gegeben sind zwei Strukturen

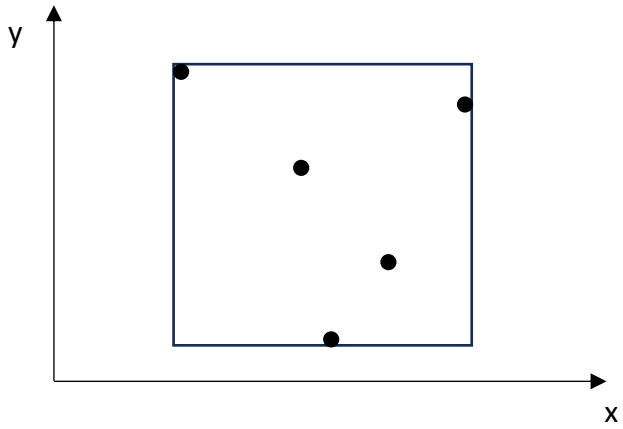
```c
#include <math.h>

typedef struct Point{
  double x,
  double y,
} Point;

typedef struct Rectangle{
  Point lowerLeftCorner,
  Point upperRightCorner,
} Rectangle;
```



a) Schreiben Sie eine Funktion namens `is_square(Rectangle *rectangle)`, die überprüft, ob es sich bei dem übergebenen Rechteck um ein Quadrat handelt. Falls es sich um ein Quadrat handelt wird 1 zurückgegeben, falls nicht 0.

Antwort:

12.b) Vervollständigen Sie die Funktion `surround()`, die ein Array mit n Punkten als Eingabe übernimmt und einen Pointer auf ein Rechteck zurückgibt, das gerade groß genug ist um alle Punkte zu umfassen (siehe Zeichnung).



Das Bild zeigt ein Rechteck, das fünf gegebene Punkte eng umschließt.

Signatur der Funktion:

```
Rectangle *surround(Point points[], int n);
```

Antwort: