Klausur CDT FS12 – C Name:

Zeit: 45 Minuten

Bedingungen:

- Die Aufgaben werden auf den ausgeteilten Blättern gelöst.
- Falls der Platz nicht ausreicht, benutzen Sie bitte die Rückseiten und bringen bei der Aufgabe einen entsprechenden Verweis an.
- Blätter **nicht** auseinandernehmen!
- **Keine** elektronischen Hilfsmittel
- Wo immer möglich soll der Lösungsweg ersichtlich sein.
- Unredliches Verhalten hat die Note 1 zur Folge.

Maximale Punktzahl: 54

Erreichte Punktzahl: **Note:**

Aufgabe 1 — 6 Punkte

Was wird bei diesen Anweisungsfolgen jeweils ausgegeben?

```
double x = 3.0, y = 2.0;
printf("%f", x/y - (int)(x/y));
```

0.500000

```
enum { rot, gruen, blau };
printf("%d", blau);
```

2

```
char a='1', b[]="1";
printf("%d %d", (int)sizeof(a), (int)sizeof(b));
```

1 2 (2P)

```
int c = 5;
while (c) {
  --c;
}
printf("%d", c);
```

0

```
int punkte[] = { 5, 2, 7, 3, 3 };
printf("%d", *punkte);
```

5

Aufgabe 2 **9 Punkte**

Das folgende Programm ruft die Funktion `sumOfArrayElements()` auf. Diese bildet die Summe aller Elemente des übergebenen Arrays und gibt das Resultat zurück.

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

#define LENGTH 6

// function declaration goes here

int main(void) {
    int testArray[] = {9,8,7,6,5,4};
    (void) printf("The sum is %d\n", sumOfArrayElements(testArray, LENGTH) );
    return EXIT_SUCCESS;
}
```

Implementieren Sie die Funktion `sumOfArrayElements()`. Verwenden Sie dabei Pointer-Arithmetik und nicht die Array Notation. Beachten Sie, dass die Elemente des Arrays innerhalb der Funktion nicht verändert werden sollen.

```
int sumOfArrayElements(                                                    ) {


    for (                                                          ) {




    }


}
```

```
int sumOfArrayElements(const int *array, int arrayLength) {
    int sum = 0;
    const int *p;

    for (p = array; p < (array + arrayLength); p++) {
        sum += *p;
    }
    return sum;
}
```

parameters	2P
variables	1P
for expressions	3P
summation and return	2P
correct use of const	1P

Aufgabe 3 **a) 2 P b) 6 P c) 4 P**

a) Welcher Fehler ist dem Programmierer bei der Funktion `initPoint3D()` unterlaufen?

```
struct point3D {
    int x;
    int y;
    int z;
};

struct point3D* initPoint3D(void) {
    struct point3D p;

    p.x = 5;
    p.y = 10;
    p.z = 20;

    return &p;
}
```

```
warning: function returns address of local variable
```

b) Beschreiben Sie drei Möglichkeiten wie das Problem behoben werden kann (ohne ausprogrammieren des Codes).

```
(1) Die lokale Variable p kann als static deklariert werden

(2) Der Speicherplatz für den struct kann innerhalb der Funktion dynamisch
mit malloc() alloziert werden.

(3) Der Speicherplatz für den struct kann von der aufrufenden Funktion
alloziert werden. Danach wird ein Pointer auf den struct an initPoint3D()
übergeben.

2P pro Möglichkeit
```

c) Die folgende Variable wurde definiert und zeigt auf einen initialisierten Datensatz.

```
struct point3D* pptr;
```

Schreiben Sie ein Codefragment welches die 3 Koordinaten des Punktes auf `stdout` ausdruckt.

```
#include <stdio.h>

(void) printf("Point is %d %d %d\n", pptr->x, pptr->y, pptr->z );

(void) printf und include     1P
String mit conversion char    1P
access of struct              2P
```

Aufgabe 4 **14 Punkte**

Schreiben Sie die Funktion `copyStringToHeap()`: Die Funktion soll auf dem Heap dynamisch Speicherplatz für einen als Parameter übergebenen String allozieren, den String in diesen Speicherplatz kopieren und anschliessend den Pointer auf den neuen String zurückgeben.

Ein String ist eine mit '\0' terminierte Folge von `char`. Verwenden Sie für die Bearbeitung der Strings Funktionen aus der *Standard Library*.

```
#include


char* copyStringToHeap(                                          ) {









      return
}
```

```
#include <stdlib.h>
#include <string.h>
char* copyStringToHeap(const char *sourceString){
    int length;
    char *stringOnHeap;

    length = strlen(sourceString);
    stringOnHeap = (char *) malloc((length + 1) * sizeof(char));
    if (NULL == stringOnHeap){
        exit(1);
    }
    return strncpy(stringOnHeap,sourceString,length);
}
include 2P, Parameter mit const 2P, variables 1P, strlen 2P, malloc(+1,
sizeof, cast, assignment) 4P, strncpy and return 3P
```

Aufgabe 5 **a) 6 P b) 3 P c) 4 P**

Gegeben ist der folgende Programmcode

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void printStdoutBrief(int i);
5   void printStdoutVerbose(int i);
6   void printStdout(void (*printOut)(int i), int outVal);
7
8   static int counter;
9
10  int main(void) {
11      const int test[] = {9,8,7,6,5,4};
12      counter = 0;
13      printStdout(printStdoutVerbose,test[3]);
14      printStdout(printStdoutBrief,test[4]);
15      return EXIT_SUCCESS;
16  }
17
18  void printStdout(void (*printOut)(int i), int outVal){
19      counter++;
20      (*printOut)(outVal);
21  }
22
23  void printStdoutBrief(int i) {
24      (void) printf("%d\n", i);
25      (void) printf("%d\n", counter);
26  }
27
28  void printStdoutVerbose(int i) {
29      (void) printf("The integer is %d\n", i);
30      (void) printf("Printed %d times\n", counter);
31  }
32
```

Erklären Sie die Bedeutung der Zeilen mit den folgenden Nummern:

a) Zeile 6

Funktionsdeklaration der Funktion printStdout 1P
Rückgabewert void 1P
Parameter 1: Ein Pointer (printOut) auf eine Funktion mit einem Parameter i vom Type integer und einem Rückgabewert void 3P
Parameter 2: outVal vom Typ integer 1P

b) Zeile 8

Definition einer globalen Variable ‘counter’ vom Typ int mit dem qualifier static, d.h. sie ist nur innerhalb dieses Modules/Files sichtbar.

c) Zeile 11

Definition der automatischen Variable ‚test‘ d.h. Allozierung von Speicherplatz auf dem Stack für einen Array mit 6 konstanten ints, welche mit den Werten 9,8,7,6,5,4 initialisiert werden