BuS-Übungsgruppe 7 Übungsblatt 2 Felix Kiunke, 357322
Philipp Hochmann, 356148
Daniel Schleiz, 356092

6 Punkte

2.1

a)
`zeile[1] = 's';` 1
Diese Operation ist möglich. Es wird im char-array zeile der char an der Stelle mit Index 1, also ' ', ersetzt durch den char 's'.

b)
`zeile++;` 1
Diese Operation ist nicht möglich, da zeile als array deklariert wurde und man die Adresse des arrays auf diese Weise nicht modifizieren kann, da dies syntaktisch unkorrekt ist. Möchte man mittels solcher Pointer Arithmetik über ein array iterieren, so müsste man eine gesonderte Pointer-Variable erstellen, welche auf den ersten Eintrag des Arrays zeigt.

c)
`pi = &i;` 1
Diese Operation ist möglich. Es wird der Wert von der variable pi auf die Adresse von der Variable i im Speicher gesetzt.
Würde man pi nun mittels * dereferenzieren, so wäre der Wert von *pi = 0, also gleich dem von i.

d)
`*pi = 42;` 1
Diese Operation ist möglich. Der Wert von *pi wird also auf 42 gesetzt. Da aber der Wert von *pi an der gleichen Stelle im Speicher liegt wie der der Variable i, ist nun also auch der Wert der variable i gleich 42.

e)
`&i = pi;` 1
Diese Zuweisung ist nicht möglich. Zwar haben theoretisch &i und pi den gleichen Wert aufgrund vorheriger Operationen, jedoch ist es einfach syntaktisch nicht korrekt, einer Variable mit vorangestelltem Adressoperator etwas zuzuweisen, da sie kein "lvalue" ist.

f)
`z = &zeile[43]-i;` 1
Diese Operation ist möglich. &zeile[43] ist die Speicheradresse des Eintrags mit Index 43 im array zeile, also dem Buchstaben 't'.
Davon wird nun der Wert von i abgezogen, also 42. Jedoch arbeitet der Compiler so, dass von der Speicheradresse nicht wortwörtlich 42 subtrahiert wird, sondern er weiß, dass ein char array vorliegt, und subtrahiert deshalb von der gegebenen Speicheradresse 42 mal die Größe eines Eintrags des arrays, also je nach Architektur des auszuführenden System z.B. ein byte. Also hat &zeile[43]-i als Wert die Adresse von zeile[1]. Letzenendes wird der Variable z also &zeile[1] zugewiesen; da z ein Pointer ist könnte man z dereferenzieren, *z wäre dann gleich 's'.

BuS-Übungsgruppe 7 Übungsblatt 2 Felix Kiunke, 357322
Philipp Hochmann, 356148
Daniel Schleiz, 356092

3 Punkte

2.2

Siehe umrechnung.c, umrechnungTabelle.c

4 Punkte

2.3

a)

1

Die Ausgabe:

```
[main] : a=5, b=6
[summe]: a=1, b=2
a+b=11
[diff] : a=4, b=3
a-b=-1
```

Innerhalb der main Funktion werden die globalen Variablen a und b durch die lokalen Variablen a und b überdeckt, und da diese innerhalb der main Funktion auf 5 und 6 gesetzt werden, werden beim printen der ersten Zeile eben diese lokalen Variablen genommen, wodurch der Output 'a=5 b=6' ist und nicht 'a=1 b=2'.

In der nächsten printf Anweisung in main wird eine die Funktion summe als Parameter übergeben, weshalb diese zuerst ausgewertet werden muss. Als Parameter erhält summe die lokalen variablen a und b, weshalb die Zuweisung p1=a und p2=b stattfindet. In der Funktion selber soll wieder auf a und b zugegriffen werden zum printen von [summe]: a=1, b=2, jedoch ist die Funktion logischerweise außerhalb der main-Funktion definiert, weshalb summe als a und b die globalen Variablen nimmst, weshalb eben der Output a=1, b=2 zustande kommt. Die Funktion returnt aber die Summe von p1 und p2, welche den Wert der lokalen Variablen a und b in der main-Funktion haben, weshalb summe 5+6, also 11 returnt, was dazu führt, dass a+b=11 ausgegeben wird, da printf das zurückgegebene Ergebnis von summe verwendet.

Der Funktion diff werden wieder a und b übergeben, wodurch p1=a und p2=b zugewiesen wird, zusätzlich wird die Adresse von c übergeben. Innerhalb der Funktion diff werden lokale Variablen a und b eingeführt, welche die globalen a und b verdecken, wodurch bei der Ausgabe a=4, b=3 zustande kommt, da eben diese Zuweisungen an die neuen lokalen variablen stattfinden.
Anschließend wird p1-p2, also von den Werten her a-b mit den Werten der lokalen Variablen aus main, gerechnet, das Ergebnis wird in *d gespeichert. Da beim Funktionsaufruf die Adresse von c übergeben wurde, wird also dadurch damit auch der Wert von c überschrieben mit 6-5, also -1, was die letzte Ausgabe erklärt.

b) c)

3

Siehe sort.c, sortNew.c

7 Punkte

2.4

Siehe rot13.c

Insgesamt: 20/20, sehr schön!