EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Informatik 2
Sommersemester 2021
Prof. Dr. T. Grust, B. Dietrich, C. Duta, D. Hirn, T. Fischer

Übungsblatt 1 (28.04.2021)

Abgabe bis: Mittwoch, 05.05.2021, 14:00 Uhr

Relevante Videos bis einschließlich:
`Informatik 2 - Chapter 2 - Video #12`
https://tinyurl.com/Informatik2-SS2021

Aufgabe 1: [5 Punkte] (*Abgabe:* `arg-eval-order.c0`)

Schreibe ein C0-Programm, an dessen Ausgabe im Terminal (siehe die Funktionen in der Bibliothek `conio`) ihr eindeutig erkennen könnt, dass die Argumente e_1, e_2 und e_3 in einem Funktionsaufruf `f(`e_1`,`e_2`,`e_3`)` in der Reihenfolge e_1, e_2, e_3 ausgewertet werden. Die Funktion `f` soll hier eine beliebige Funktion mit drei Parametern sein.

Aufgabe 2: [15 Punkte] (*Abgabe:* `printbitsfit.c0`)

Schreibe eine Funktion `int printbitsfit(int x)` die, ähnlich wie die Funktion `void printbits(int x)` aus der Vorlesung, die Bits von `x` im Terminal mittels `printchar(...)` ausgibt. Dabei soll die Funktion bei der Ausgabe alle vorangehenden `0`-Bits unterdrücken. Der Rückgabewert der Funktion gibt die Anzahl der Zeichen an, die zur Darstellung von `x` im Terminal ausgegeben wurden. Die Funktion `printbitsfint(...)` hat also einen Seiteneffekt und einen Rückgabewert.

Baut in eure `main()`-Funktion mindestens drei Funktionsaufrufe zu `printbitsfit(...)` ein, die die Funktionsweise der Funktion deutlich machen.

Assignment Project Exam Help

Hinweis: Verwendet die aus der Vorlesung bekannte Funktion `bool is_bit_set(int x, int n)`.

Beispiele: Hier sind einige Beispielaufrufe für `printbitsfit(...)`:

https://tutorcs.com

```
--> printbitsfit(0); printchar('\n');
1 (int)
0
(void)
--> printbitsfit(42); printchar('\n');
6 (int)
101010
(void)
--> printbitsfit(5249); printchar('\n');
13 (int)
1010010000001
(void)
```

WeChat: cstutorcs

Aufgabe 3: [7 Punkte] (*Abgabe:* `bitwidth.c0`)

Schreibe eine Funktion `int bitwidth()`. Diese Funktion stellt fest, wieviele Bits die Sprache C0 intern zur Darstellung von Werten des Typs `int` einsetzt (derzeit müsste die Funktion also das Ergebnis `32` liefern).

Schreibt die Funktion unter der Annahme, dass sich in Zukunft die Anzahl der Bits für den Typ `int` ändern könnte.

Baut eure `main()`-Funktion so, dass beim Ausführen des übersetzten Programms einfach der Rückgabewert von `bitwidth()` auf dem Terminal ausgegeben wird.

Hinweis: Nutzt dazu die Bit-Operatoren, die wir in der Vorlesung kennen gelernt haben. Eine Implementation der Form `return 32;` ist genau 0 Punkte wert. ☺

Aufgabe 4: [7 Punkte] (*Abgabe:* `table.c0`)

Schreibe ein C0-Programm, das eine Tabelle auf dem Terminal ausgibt, die für jede Kombination der Werte `a` $\in \{-10, -9, -8, \ldots, 0, \ldots, 8, 9, 10\}$ und `b` $\in \{-5, -3, -1, 1, 3, 5\}$ die folgenden fünf Einträge enthält:

```
a    b    a/b    a%b    (a/b)*b+(a%b)
```

Hier seht ihr einen Ausschnitt aus der Tabelle die das Programm ausgeben soll:

```
a       b       a/b     a%b     (a/b)*b+(a%b)
-10     -5      2       0       -10
-10     -3      3       -1      -10
-10     -1      10      0       -10
-10     1       -10     0       -10
-10     3       -3      -1      -10
-10     5       -2      0       -10
-9      -5      1       -4      -9
...
10      5       2       0       10
```

Hinweis: Die Ausgabe eines Tabulators mittels `printchar('\t');` zwischen den Einträgen einer Zeile hilft bei der Formatierung der Tabelle.

Aufgabe 5: [6 Punkte] (*Abgabe:* `huh.txt`)

Bei der Suche in alten Ordnern auf eurem Rechner findet ihr ein C0-Programm vor, das folgende *unkommentierte* Funktion definiert:

Assignment Project Exam Help

```
1  int huh(int x) {
2    int c = 0;
3
4    while (x != 0) {
5      c = c + 1;
6      x = x & x - 1;
7    }
8
9    return c;
10 }
```

https://tutorcs.com

WeChat: cstutorcs

1. [3 Punkte] Wendet `huh` auf eine Reihe von Argumenten an, um eine Vermutung aufzubauen, *was* die Funktion leistet. Erläutert eure Vermutung kurz in eigenen Worten. (max. 280 Zeichen) und speichert diese im Plain-Text-File `huh.txt`.
2. [3 Punkte] **Achtung:** Knifflige Teilaufgabe! Erklärt im Detail und so exakt wie möglich, *wie* und *wieso* `huh` überhaupt funktioniert. Fügt auch diese Erklärung dem File `huh.txt` hinzu.

   Hinweis: Geht bei eurer Erklärung *insbesondere* auf die Zeile 6 im Code ein.