64-041 Übung Rechnerstrukturen WS 15/16

Aufgabenblatt 5 Ausgabe: 11.11., Abgabe: 18.11. 24:00

Gruppe	
Name(n)	Matrikelnummer(n)

Aufgabe 5.1 (Punkte 10+10+5)

ULP: Ihr Kollege möchte die Rundungsfehler bei Gleitkomma-Arithmetik besser verstehen und hat Ihnen folgendes Programm geschickt:

```
public class auf_5_1 {

  public static void main( String args[] ) {
    // calculate the sum of 1E8 unprecise numbers
    int n = 0;
    float sum;
    float limit = 1.0E8f;
    for( sum = 0; sum < limit; sum += 0.1 ) {
      n++;
      // if ((n % 100000) == 0) {
      //   System.out.println( "n=" + n + " sum=" + sum + " target=" + (n*0.1));
      // }
    }
    System.out.println( "sum is " + sum + " compared to " + (n*0.1) );
  }
}
```

(a) Das Programm besteht aus einer Schleife, die immer wieder den Wert 0.1 (der sich in binärer Gleitkommadarstellung nicht exakt darstellen lässt) auf die Variable `sum` addiert. Was erwarten Sie (ungefähr) als Ausgabewert des Programms?

(b) Was passiert tatsächlich? Warum? (Es kann helfen, die auskommentierten Codezeilen wieder zu aktivieren).

(c) Schreiben Sie das Programm so um, dass es den ursprünglich angedachten Zweck erfüllt.

Aufgabe 5.2 (Punkte 10+5)

ASCII-Code (ISO-8859-1): Entschlüsseln Sie mit Hilfe der Tabellen aus dem Vorlesungsskript (Zeichensatz nach ISO-8859-1) die folgende Zeichenkette. Schreiben Sie dabei auch alle Steuerzeichen mit auf. Die einzelnen Zeichen sind als Hexwerte angegeben.

(a) `78 6b 63 64 20 69 73 74 20 65 69 6e 20 57 65 62 63 6f`
`6d 69 63 20 76 6f 6e 20 52 61 6e 64 61 6c 6c 20 4d 75`
`6e 72 6f 65 2e 0a`

(b) Auf welcher „Art“ Rechner wurde der Text erstellt?

Aufgabe 5.3 (Punkte 10+5+5)

UTF-8: Die ISO-8859-1 Kodierung benutzt 8 Bit für jedes enthaltene Zeichen. Die direkte Kodierung der basic-multilingual Plane von Unicode (Java Datentyp `char`) verwendet pro Zeichen 16 Bit, während die UTF-8 Kodierung Vielfache von 8 Bit benutzt.

(a) Wir betrachten einen deutschsprachigen Text mit insgesamt 100 000 Zeichen. Wir nehmen die folgenden Wahrscheinlichkeiten für die Umlaute an, andere Sonderzeichen kommen nicht vor:

Ä/ä	Ö/ö	Ü/ü	ß
0,56%	0,29%	0,62%	0,31%

Wie viele Bytes belegt dieser Text bei Kodierung nach ISO-8859-1, in direkter Unicode Darstellung und in UTF-8?

(b) Wir betrachten einen chinesischen Text mit insgesamt 100 000 Schriftzeichen. Im Unicode-Standard sind für die CJK-Symbole (chinesisch, japanisch, koreanisch) die Bereiche von `U+3400` bis `U+4DBF` und `U+4E00` bis `U+9FCF` reserviert.

Wie viele Symbole sind das?

(c) Wie viele Bytes belegt der chinesische Text bei direkter Unicode Darstellung und bei Kodierung als UTF-8?

Aufgabe 5.4 (Punkte 5+5)

Shift-Operationen statt Multiplikation: Ersetzen Sie die folgenden Berechnungen *möglichst effizient* durch eine Folge von Operationen: <<, +, -. Nehmen Sie für die Variablen x und y den Datentyp `int` (32-bit Zweierkomplementzahl) an.

(a) $y = 12 * x$

(b) $y = -56 * x$

Aufgabe 5.5 (Punkte 5+5+10+15)

Logische- und Shift-Operationen: Realisieren Sie, die folgenden Funktionen als *straightline*-Code in Java, das heißt ohne Schleifen, If-Else Abfragen oder den ternären Operator `.. ? .. : ...` Außerdem dürfen nur einige der logischen und arithmetischen Operatoren benutzt werden:

```
! ~ & ^ | + << >> >>>
```

Alle Eingabeparameter und Rückgabewerte sind jeweils (32-bit) Integerwerte.

(a) `bitNand(x,y)` Diese Funktion soll das bitweise NAND liefern: $\overline{x_i \wedge y_i}$. Als Operatoren dürfen nur | und ~ (OR, Negation) benutzt werden.

(b) `bitXNor(x,y)` Diese Funktion soll die XNOR-Verknüpfung (Äquivalenz) realisieren: $x_i = y_i$. Als Operatoren dürfen nur & und ~ (AND, Negation) benutzt werden.

(c) `base64(b1,b2,b3)` Schreiben Sie eine Funktion, die die jeweils unteren 8 bit der übergebenen drei Integerparameter in die 32 bits der zugehörigen base64-Kodierung umsetzt.

(d) `abs(x)` Der Absolutwert (Betrag) von `x`. Welchen Wert liefert ihre Funktion für den Eingabewert -2^{31}? Beschreiben Sie, wie Ihre Lösung funktioniert.