Aufgabenblatt 4

Ausgabe 09/11/2009, Abgabe bis 16/11/2009 12:00

Name(n):
Matrikelnummer(n):
Übungsgruppe:

Aufgabe 4.1 Unicode (10 Punkte)

Im Unicode-Standard sind für die CJK-Symbole (chinesisch, japanisch, koreanisch) die Bereiche von `U+3400` bis `U+4BDF` und `U+4E00` bis `U+9FCF` reserviert. Wie viele Symbole sind das?

Führen Sie die Rechnung im Hexadezimalsystem aus und verwenden Sie dabei das 16-Komplement für die Subtraktion. Bitte die Lösung mit Zwischenschritten abgeben.

Aufgabe 4.2 Shift-Operationen statt Multiplikation (10 Punkte)

Geben Sie eine Folge von Java-Operationen an, mit denen man die folgenden Berechnungen effizient durch Shifts und Addition ersetzen könnte. Nehmen Sie für die Variablen `x` und `y` den Datentyp `int` (32-bit Zweierkomplementzahl) an.

a) $y = 20 * x$

b) $y = 30 * (x + 2)$

Aufgabe 4.3 Rotate-Operationen (10+10 Punkte)

In der Vorlesung wurden die rotate-left und rotate-right Operationen vorgestellt, die aber in Java und C nicht als Operatoren definiert sind. Schreiben Sie daher eine Java-Klasse, die die rotate-Operationen als Methoden zur Verfügung stellt:

```
public static int rotateLeft( int i, int distance ) {
  return ...;
}

public static int rotateRight( int i, int distance ) {
  return ...;
}
```

Versuchen Sie, diese beiden Funktionen in Ihrer Klasse mit logischen und Shift-Operationen zu implementieren. Eine Vorlage inklusive Selbsttest finden Sie als Java-Klasse `RotateLeftRight` im `examples`-Unterverzeichnis der Vorlesungs-Webseite.

Hinweis: die beiden obigen Methoden sind natürlich in der Java-Klassenbibliothek in der Klasse `java.lang.Integer` enthalten. Hier geht es darum, deren Funktion zu verstehen.

Aufgabe 4.4 Logische Operationen (5+5+10+20+20 Punkte)

Versuchen Sie, die folgenden Operationen als *straightline*-Code in Java zu realisieren, das heisst ohne Schleifen oder If-Else Abfragen. Außerdem dürfen nur die folgenden logischen und arithmetischen

Operatoren benutzt werden:

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

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

a) `bitNor(x,y)` (5 Punkte)

Diese Funktion soll das bitweise NOR von `x` und `y` liefern. Als Operatoren dürfen nur `&` und `~` (AND, Negation) benutzt werden.

b) `bitXor(x,y)` (5 Punkte)

Diese Funktion soll die XOR-Verknüpfung von `x` und `y` liefern. Als Operatoren dürfen nur `&` und `~` (AND, Negation) benutzt werden.

c) `getByte(x,n)` (10 Punkte)

Diese Funktion soll das angegebene Byte ($0 \leq$ `n` ≤ 3) aus dem Wert `x` extrahieren.

d) `bitCount(x)` (20 Punkte)

Diese Funktion liefert die Anzahl der 1-Bits in `x`.

Hinweis: Es gibt tatsächlich elegante und effiziente Lösungen ohne Schleifen. Volle Punktzahl nur für eine Lösung, die mit weniger als 32 Operationen auskommt.

e) `abs(x)` (20 Punkte, tricky)

Der Absolutwert (Betrag) von `x`. Keine If-Abfragen, nur logische und Shift-Operationen verwenden. Welchen Wert liefert ihre Funktion für den Eingabewert -2^{31}? Beschreiben Sie, wie Ihre Lösung funktioniert.