TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen
Praktikum Grundlagen der Programmierung
Prof. Dr. Helmut Seidl

WS 2007/2008
Aufgabenblatt 2
26. Oktober 2007

Abgabe: Montag, 05.11.07, 8:00Uhr, per Mail/Papier beim jeweiligen Tutor

Praktikum Grundlagen der Programmierung

Aufgabe 8 (Ü) Erweiterter Euklid

Der erweiterte euklidsche Algorithmus ist ein Algorithmus, der den größten gemeinsamen Teiler (ggT) von zwei Zahlen m,n bestimmt und zusätzlich Koeffizienten u,v, so dass $ggT(m,n) = m \cdot u + n \cdot v$. Implementieren Sie den Euklid-Algorithmus zur Berechnung des ggTs, so dass er für alle $m,n \in \mathbb{N}^+$, $m \leq n$ ganze Zahlen $u,v \in \mathbb{Z}$ mit $ggT(m,n) = m \cdot u + n \cdot v$ berechnet. Zum Beispiel werden für $m = 13$ und $n = 17$ die Zahlen $u = 4$ und $v = -3$ berechnet.

Aufgabe 9 (Ü) 17 und 4

Hier wird ein Computerspiel entwickelt, welches auf dem französischen Kartenspiel *17 und 4* (bekannt als *Black Jack*) basiert. Die Regeln des Spieles sind folgende:

> Zwei Spieler spielen gegen einander. Ziel des Spiels ist es, mit zwei oder mehr Karten näher an 21 Punkte heranzukommen als der andere, ohne dabei den Wert von 21 Punkten zu überschreiten. Eine Karte kann einen Wert von 2 bis 11 Punkten haben.
> Jeder Spieler hat am Anfang zwei Spielkarten. Er kann nun entscheiden, ob er weitere Karten ziehen möchte oder nicht. Glaubt er, nahe genug an 21 Punkte herangekommen zu sein, so lehnt er weitere Karten ab. Wenn er durch einen Zug 22 oder mehr Punkte erreicht, verliert er sofort.
> Es gewinnt der Spieler, der als erster am nähesten an 21 Punkte herankommt.

Schreiben Sie ein Mini-Java Programm `SuV.java`, mit dem man *17 und 4* zu zweit spielen kann. Jeder Spieler soll über Dialogboxen gefragt werden, ob er weitere Karten ziehen will: 1 für ja, 0 für nein.

Hinweis: Sie können für diese Aufgabe zusätzlich zu `MiniJava` die Klasse `BlackJackMiniJava` von der Übungshomepage herunterladen, die Ihnen die statische Methode `int drawCard()` zum Ziehen von Karten anbietet.
Ersetzen Sie einfach `extends MiniJava` durch `extends BlackJackMiniJava`.

Aufgabe 10 (Ü) Syntax-Baum

Zeichnen Sie für das folgende MiniJava-Programm den Syntax-Baum. Dazu steht Ihnen die Grammatik von MiniJava im Anhang zur Verfügung.

```
int x,r,n;
r = 1;
n = 1;
x = read();
while (n < x) {
  if (r % 1 == 0)
      r = r * n;
  else
      r = r * (-n);
  n = n + 1
  write (r);
}
```

Aufgabe 11 (H) Primfaktorzerlegung (4 Punkte)

Jede natürliche Zahl $n \geq 0$ ist entweder selbst eine Primzahl oder lässt sich als ein bis auf die Reihenfolge eindeutiges Produkt von Primzahlen darstellen (Primfaktorzerlegung). Zur Berechnung der Primfaktorzerlegung einer Zahl *n* sollen Sie der Reihe nach durch 2 und durch die ungeraden Zahlen ab der 3 teilen. Wenn ein Teiler gefunden wurde, so wird dieser ausgegeben und der Rest in Primfaktoren zerlegt.

Schreiben Sie ein MiniJava-Programm, das eine natürliche Zahl $n \geq 0$ einliest und in Primfakoren zerlegt. Ihr Programm soll alle Primfaktoren der Zahl *n* am Bildschirm ausgeben. Zum Beispiel ergibt sich für die Zahl 24 folgende Zerlegung: $2 \cdot 2 \cdot 2 \cdot 3$.

Aufgabe 12 (H) Meiern (6 Punkte)

Wir entwickeln hier ein Computerspiel, basierend auf dem bekannten Würfelspiel *Meiern*. Die Regeln des Spieles sind folgende:

> Für das Spiel braucht man 2 Würfel. Ein Spieler tritt gegen den Computer an. In jeder Spielrunde würfelt der Spieler mit beiden Würfeln. Die beiden gewürfelten Zahlen kombiniert er zu einer zweistelligen Zahl, so dass die größere von beiden als 10er Stelle und die kleinere von beiden als 1er Stelle interpretiert wird. Die Reihenfolge der Würfe ist die folgende:
> Der höchste Wurf, der sogenannte *Meier (21)* ist durch keinen anderen Wurf zu überbieten. Ein Spieler, der einen *Meier* würfelt hat sofort gewonnen. Kurz nach dem *Meier* folgt das *6er-Pasch (66)*, gefolgt von den anderen Paschen in absteigender Reihenfolge. Auf das kleinste Pasch *(11)* folgen die normalen Würfe in absteigender numerischer Reihenfolge von *(65)* bis hinab zum kleinstmöglichsten Wurf *(31)*.
> Es wird solange abwechselnd von Computer und Spieler gewürfelt, bis der aktuelle Spieler den Wurf des vorigen Spielers nicht mehr überbieten kann, und das Spiel verliert.

Schreiben Sie ein Mini-Java Programm, mit dem man *Meiern* gegen den Computer spielen kann. Der Spieler soll abwechselnd über Dialogboxen nach seinen Würfen gefragt werden, und als Reaktion darauf die Würfe des Computers mitgeteilt bekommen.

Hinweis: Sie können für diese Aufgabe zusätzlich zu `MiniJava` die Klasse `WuerfelnMiniJava.java` von der Übungshomepage herunterladen, die Ihnen die statische Methode `int rollDice()` zum Würfeln anbietet. Ersetzen Sie einfach `extends MiniJava` durch `extends WuerfelnMiniJava`.

Aufgabe 13 (H) Syntax-Baum **(3 Punkte)**

Zeichnen Sie für das folgende MiniJava-Programm den Syntax-Baum. Dazu steht Ihnen die Grammatik von MiniJava im Anhang zur Verfügung.

```
int a,b,c,d;
a = read();
b = read();
c = -1;
d = 9;
while (a < b) {
  a = a + 1;
}
if (a > c) {
  if (a < d) {
      write(a);
  }
}
```

Anhang

Die Grammatik von MiniJava (aus der Vorlesung):

<program> ::= <decl>* <stmt>*
<decl> ::= <type> <name> (, <name>)* ;
<type> ::= int
<stmt> ::= ; | { <stmt>* } | <name> = <expr>; | <name> = read(); | write(<expr>); |
if (<cond>)<stmt> | if (<cond>)<stmt> else <stmt> | while (<cond>) <stmt>
<expr> ::= <number> | <name> | (<expr>) | <unop> <expr> | <expr> <binop> <expr>
<unop> ::= -
<binop> ::= - | + | * | / | %
<cond> ::= true | false | (<cond>) | <expr> <comp> <expr> | <bunop> <cond> |
<cond> <bbinop> <cond>
<comp> ::= == | != | <= | < | >= | >
<bunop> ::= !
<bbinop> ::= && | ||