Friedrich-Schiller-Universität Jena **Fakultät für Mathematik und Informatik** **Institut für Informatik** **Lehrstuhl für Programmiersprachen und Compiler**	**Höhere Programmierung** **SS 2001**	**Aufgabenblatt 11** **Ausgabe: 11.06.2001** **Abgabetermin:** **18.06.2001 16:00 Uhr**

Aufgabe 1: Verweise

Definieren Sie die Verweistypen für die folgenden Zieltypen. Definieren Sie jeweils eine zugehörige Verweisvariable mit nichttrivialem Wert.

a)
```
SUBTYPE IntegerSub IS Integer RANGE 0..100;
```

b)
```
Float
```

c)
```
TYPE ArrayTyp IS ARRAY (Integer RANGE <>) OF Character;
```

d)
```
TYPE XYZTyp(A: Boolean) IS RECORD
  B: Integer;
  CASE A IS
    WHEN True =>
      C: Float;
      D: String(1..20);
    WHEN FALSE =>
      E: Integer;
      F: String(1..30);
  END CASE;
END RECORD;
```

e)
```
TYPE RecordTyp IS RECORD
  A, B: Integer;
  C: ???; -- Die Komponente C soll auf Objekte des Typs RecordType
          -- verweisen können
END RECORD;
```

f) Verweistyp von **d)**

5 Punkte

Aufgabe 2: Verweise

Geben Sie an, was die folgenden Programmfragmente ausgeben. Begründen Sie ihre Antwort.

a)
```
TYPE IntegerAccessTyp IS ACCESS Integer;
Int: Integer;
AccessInt1, AccessInt2: IntegerAccessTyp;

BEGIN
  AccessInt1 := NEW Integer(5);
  AccessInt2 := AccessInt1;
  AccessInt1.all := AccessInt2.all+1;
  Ada.Integer_Text_Io.Put(AccessInt2.all);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(AccessInt1.all);
  Ada.Text_Io.New_Line;
END;
```

b)
```
TYPE ListenTyp;
TYPE ListenAccess is ACCESS ListenTyp;
TYPE ListenTyp IS RECORD
  A: Integer;
  B: ListenAccess:=null;
END RECORD;
Liste1: ListenAccess := new ListenTyp; --(1)
Liste2: ListenAccess;

BEGIN
  Liste1.all.A := 0;
  Liste1.all.B := new ListenTyp;
  Liste1.all.B.all.A := 1;
  Liste1.all.B.all.B := new ListenTyp;
  Liste2 := Liste1.all.B.all.B;
  Liste1:=Liste1.all.B;

  Ada.Integer_Text_Io.Put(Liste1.all.A);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(Liste2.all.A);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(Liste2.all.B.all.A);
  Ada.Text_io.New_Line;
  --(2)

END;
```

Kann in Zeile (2) noch auf das in Zeile (1) erzeugte Listenelement zugegriffen werden?

c)
```
TYPE IntegerAccessType IS ACCESS Integer;
SUBTYPE IntegerSubType IS Integer RANGE 1..100;
Zeiger1: IntegerAccessType := NEW IntegerSubType'(99);
Zeiger2: IntegerAccessType;
TauschZeiger: IntegerAccessType;

BEGIN
  Zeiger2 := NEW Integer'(55);
  TauschZeiger := Zeiger1;
  Zeiger1      := Zeiger2;
  Zeiger2      := TauschZeiger;

  Ada.Integer_Text_Io.Put(Zeiger1.all);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(Zeiger2.all);
  Ada.Text_Io.New_Line;

  TauschZeiger.all := Zeiger2.all;
  Zeiger2.all := Zeiger1.all;
  zeiger1.all := TauschZeiger.all;

  Ada.Integer_Text_Io.Put(Zeiger1.all);
  Ada.Text_Io.New_Line;
  Ada.Integer_Text_Io.Put(Zeiger2.all);
  Ada.Text_Io.New_Line;

END;
```

6 Punkte

Aufgabe 3: Listen- und Baumstrukturen

Entwickeln Sie die folgenden Strukturen mit Verweistypen.

a) Doppelt verkettete Liste mit Ringstruktur wie sie in Abbildung 1 gezeigt wird. Der Wert soll mit nichttrivialen Floattyp-Zahlen in aufsteigender Reihenfolge initialisiert werden.

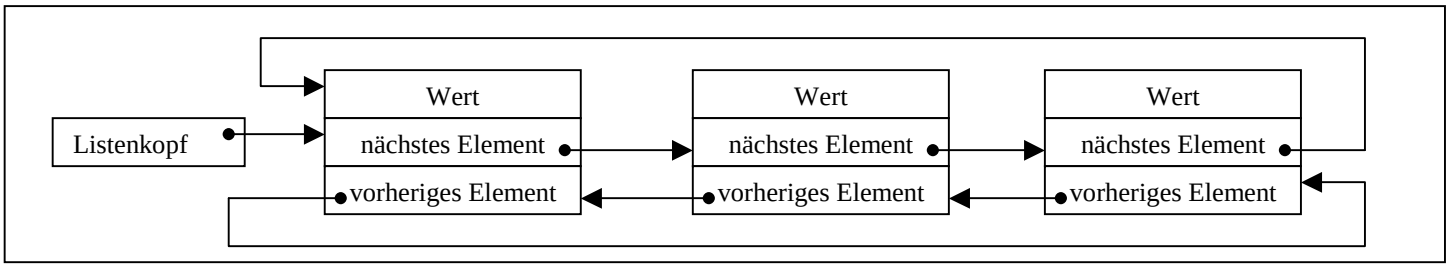

Abbildung 1: Doppelt verkettete Liste

b) Binäre Baumstruktur wie sie in Abbildung 2 gezeigt wird. Der Wert soll die entsprechende Wahrscheinlichkeit für den Knoten als Floatzahl angeben.

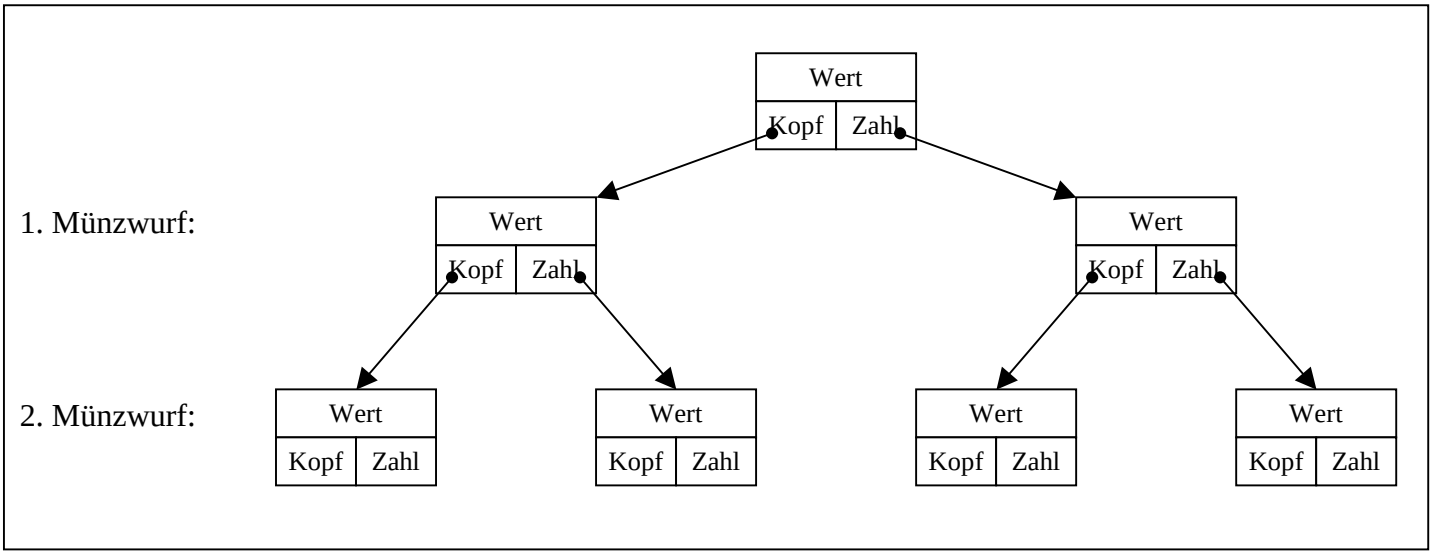

Abbildung 2: Wahrscheinlichkeitsbaum für zwei Münzwürfe

4 Punkte

Aufgabe 4: Sortieren mit einer Liste (Zusatzaufgabe für Wirtschaftsinformatiker)

Entwickeln Sie ein Programm, das nacheinander Zeichenketten mit einer Maximallänge von 100 Zeichen einliest und diese dann in einer Liste in aufsteigender Reihenfolge einsortiert. Die Eingabe soll mit einem leeren String beendet werden und die sortierte Liste soll dann ausgegeben werden.

5 Punkte