Aufgabe 1 (6 Punkte)

Gegeben ist die folgende Funktion `WasPassiert`.

```
function WasPassiert(a:integer):boolean;

  var
  b:integer;
  c:integer;

begin
  b := 1;
  c := 1;
  while (a > b) do
  begin
    c := c + 2;
    b := b + c;
  end;
  WasPassiert := (a = b);
end;
```

Geben Sie eine Problemspezifikation an, die durch die Funktion `WasPassiert` gelöst wird. Wählen Sie dabei den Wertebereich der Eingabe so allgemein wie möglich.

Eingabe: $z \in \mathbb{Z}$

Ausgabe: $b \in \{true, false\}$

Nachbedingung: $b = true \Leftrightarrow z$ ist (positive) Quadratzahl

Aufgabe 2 (6 Punkte)

Gegeben sind die folgenden Typendefinitionen, die Funktionen `c, m, f` und die drei Listen `X, Y, Z`.

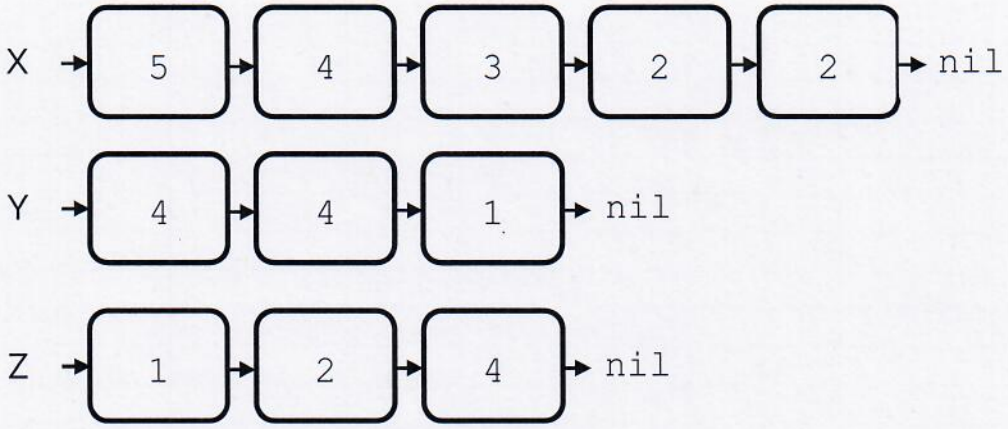
```
type
tRefListe = ^tListe;
tListe = record
           wert: integer;
           next: tRefListe
         end;

function c(inA:tRefListe;inB:tRefListe):boolean;
begin
  if ((inA = nil) or (inB = nil)) then
    c := ((inA = nil) and (inB = nil))
  else
    c := c(inA^.next, inB^.next)
end;

function m(inA:tRefListe;inB:tRefListe):integer;
begin
  if ((inA = nil)) then
    m := 0
  else
    m := inA^.wert * inB^.wert + m(inA^.next,inB^.next)
end;

function f(inA:tRefListe;inB:tRefListe):integer;
begin
  if c(inA,inB) then
    f := m(inA,inB)
  else
    f := 0
end;
```

Name: _____ Matrikelnr.: _____



Welche Ergebnisse liefern die folgenden zwei Aufrufe?

`f(X,Y)` 0

`f(Y,Z)` 16

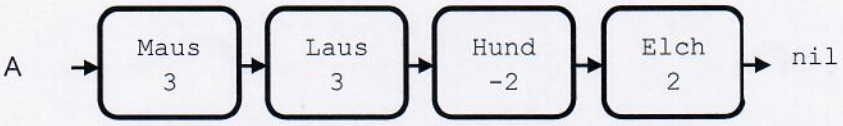
Aufgabe 3 (6 Punkte)

Gegeben sind die folgenden Typendefinitionen für eine Liste aus Zeichenketten mit einem zusätzlichen Wert `anzahl` und die Prozedur `add`.

```
tRefListe = ^tListe;
tListe = record
           text:String;
           anzahl:integer;
           next:tRefListe
         end;
```

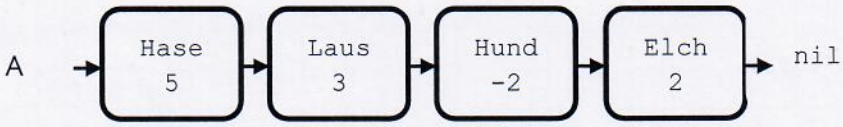
Die Prozedur `add` erfüllt die folgende Aufgabe: Für eine Liste, eine Zeichenkette und eine Anzahl, durchsucht die Prozedur die Liste. Existiert bereits ein Listenelement mit der gleichen Zeichenkette, wird die Anzahl dieses Elements um den eingegebenen Wert erhöht. Wird die Anzahl dadurch `0`, wird das Element aus der Liste ausgekettet. Existiert kein Listenelement mit der gleichen Zeichenkette, wird ein neues Element an den Anfang der Liste eingefügt. Gehen Sie davon aus, dass jede Liste jede Zeichenkette nur einmal enthält.

Hier sehen Sie Beispiele, wie die Prozedur `add` eine Liste `A` verändert:

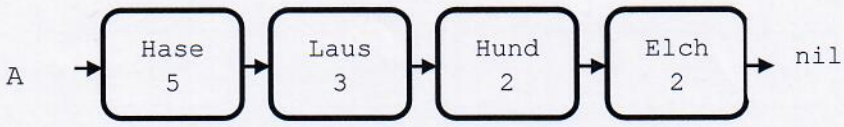



```
add(A,'Hase',5)
```


```
add(A,'Maus',-3)
```




```
add(A,'Hund',4)
```



Name: _____ Matrikelnr.: _____

Hier finden Sie die Prozedur add. Die Prozedur ist jedoch noch unvollständig. Ergänzen Sie diese passend, an den grau eingefärbten Stellen, jeweils um eine Zeile.

```
procedure add(var ioSet:tRefListe; inText:String; inZahl:integer);

  var
  neu: tRefListe;
  lauf: tRefListe;
  lauf2: tRefListe;
  found: boolean;

  begin
    lauf := ioSet;
    lauf2 := lauf;
    found := false;
    while (lauf <> nil) do
    begin
      if (lauf^.text = inText) then
      begin
        lauf^.anzahl := lauf^.anzahl + inZahl;
        found := true;
        if (lauf^.anzahl = 0) then
        begin
          if (lauf2 = lauf) then
            ioSet := ioSet^.next
          else
          begin
            lauf2^.next := lauf^.next
          end
        end
      end;
      lauf2 := lauf;
      lauf := lauf^.next;
    end;

    if (not found) then
    begin
      new(neu);
      neu ^.text := inText;
      neu ^.anzahl := inZahl;
      neu ^.next := ioSet;
      ioSet := neu;
     end
  end;
```

Aufgabe 4 (6 Punkte)

Eine Funktion gibt für eine nicht-negative ganze Zahl aus, welcher Rest verbleibt, wenn die Zahl (ganzzahlig) durch 3 geteilt wird. Sie soll einem funktionsorientierten Test unterzogen werden.

Bekannt sind die Typdefinitionen sowie der Funktionskopf:

```
type
tZahl = 0..maxint;
tRest = 0..2;

function divDrei(inZahl:tZahl):tRest;
{berechnet den Wert inZahl % 3}
```

Geben Sie für einen Black-Box-Test eine Zerlegung der Menge der zulässigen Eingabedaten in sechs sinnvolle Äquivalenzklassen an. Benennen Sie hierbei zusätzlich zu jeder Äquivalenzklasse beispielhaft ein konkretes Testdatum.

Klasse	Datum
• Rest 0, kleiner 3	(0,0)
• Rest 1, kleiner 3	(1,1)
• Rest 2, kleiner 3	(2,2)
• Rest 0, größer gleich 3	(3,0)
• Rest 1, größer gleich 3	(4,1)
• Rest 2, größer gleich 3	(5,2)