Aufgabe 1

a) Das Objektmodell aus Abbildung 2 der Aufgabenstellung enthält zwei Fehler.

Der erste Fehler betrifft die drei Objekte Punkt 2, Punkt 3 und Punkt 4: Die Klasse *Punkt* ist eine Unterklasse der abstrakten Klasse *GeoObjekt*, erbt also insbesondere die Behälterbeziehung zur Klasse *Ebene*. Die Behälterbeziehung schreibt aber durch ihre Ordnung zwingend vor, das jedes Punkt-Objekt in genau einem Objekt der Klasse *Ebene* enthalten ist. Die angesprochenen Punktobjekte erfüllen diese Anforderung nicht. Wir ergänzen daher das Objektmodell um je eine gerichtete Objektbeziehung zwischen dem Behälterobjekt Ebene 1 und seinen Teilobjekten Punkt 2, Punkt 3 und Punkt 4 (Abbildung 1.1).

Der zweite Fehler betrifft das Objekt Punkt 3. Dieses Objekt ist konstruktiver Bestandteil der Objekte Segment 1 und Segment 2. Die Ordnung der Konstruktionsbeziehung zwischen Punkt- und Segmentobjekten schreibt jedoch vor, daß ein Punkt höchstens Teilobjekt eines Segments sein darf. Zur Korrektur ergänzen wir das Objektmodell um den Punkt 3.1, der eine Kopie von Punkt 3 ist und als Teilobjekt von Segment 2 eingefügt wird (Abbildung 1.1).

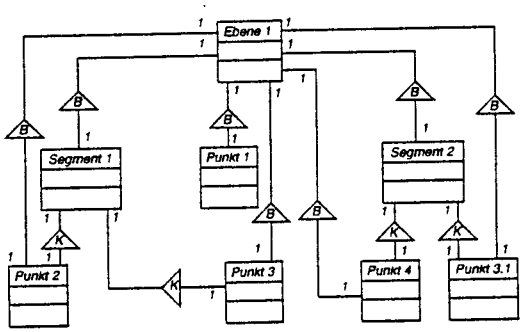

Abbildung 1.1: Ein zulässiges Objektmodell.

b) Bei der Besprechung der Teilaufgaben (b) und (c) gehen wir davon aus, daß zu Beginn des datenstrukturorientierten Grobentwurfs alle Analyseklassen in entsprechende ADT-Moduln transformiert worden sind.

In Teilaufgabe (b) ist die Vererbungsbeziehung zwischen der abstrakten Klasse *GeoObjekt* und ihren konkreten Unterklassen *Punkt* und *Segment* nach der ersten Alternative zu transfor-

mieren, d.h. die entsprechenden UnterklassenADTs Punkt und Segment benutzen den OberklassenADT GeoObjekt.

Als nächstes übertragen wir die Konstruktionsbeziehung zwischen den Objekten der Klasse *Punkt* und Objekten der Klasse *Segment*. Da die Ordnung der gerichteten Objektbeziehung nach oben beschränkt ist und die einzelnen (Punkt-) Teilobjekte des Segments als Start- bzw. Endpunkt unterscheidbar sind, resultiert die Transformation dieser Konstruktionsbeziehung in einer einfachen Benutzungsbeziehung zwischen den ADT- Moduln Segment und Punkt. Eine dynamische Datenstruktur zur Verwaltung der Beziehung wird hier also nicht benötigt.

Abschließend betrachten wir die Behälterbeziehung zwischen Objekten der Klasse *GeoObjekt* und Objekten der Klasse *Ebene*. Da die Klasse *GeoObjekt* abstrakt ist, kann eine Ebene nur Punkt- oder Segmentobjekte enthalten. Punkt- oder Segmentobjekte sind stets Teilobjekte von genau einer Ebene, während die Anzahl der Punkt- bzw. Segmentobjekte in einer Ebene nach oben unbeschränkt ist. Eine Ebene benötigt daher zwei dynamische Datenstrukturen zur Verwaltung ihrer Punkt- bzw. Segmentteilobjekte, die wir aus dem GADT-Modul Objektliste instantiieren.

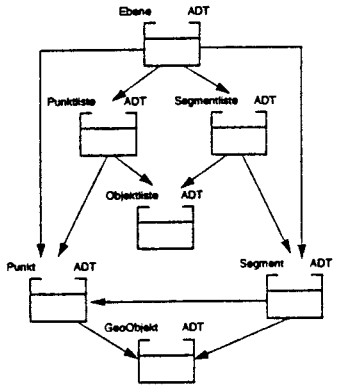

Abbildung 1.2: Transformation nach der 1. Alternative.

c) Transformieren wir die Vererbungsbeziehung nach der zweiten Alternative, benutzt — anders als in Teilaufgabe (b) — der OberklassenADT GeoObjekt die UnterklassenADTs Punkt und Segment und exportiert die Operationen der UnterklassenADTs. Auf Typebene kann jetzt nicht mehr zwischen Segmenten und Punkten unterschieden werden. Damit diese notwendige Unterscheidung weiterhin möglich ist, müssen vom ADT-Modul GeoObjekt entsprechende Operationen zur Verfügung gestellt werden (Abbildung 1.3).

Die Transformation der Konstruktionsbeziehung zwischen den Objekten der Klasse *Punkt* und Objekten der Klasse *Segment* ändert sich im Vergleich zur Teilaufgabe a) nicht. Da die Vererbungsstruktur aber hier nach der zweiten Alternative transformiert wurde, benötigt eine Ebene nur noch eine einzige dynamische Datenstruktur, denn zwischen Objekten der Klasse *Punkt*

bzw. *Segment* kann in diesem Fall nur durch entsprechende Operationen des ADT-Moduls GeoObjekt unterschieden werden.

Durch die Anwendung der zweiten Transformationsalternative sind im Vergleich zur Teilaufgabe (b) einige Benutzungsbeziehungen und ADT-Moduln entfallen. Diese „Vereinfachung" des Komponentengraphen wird durch die gezielte Abschwächung des Typkonzepts sowie zusätzlicher Funktionalität erkauft.

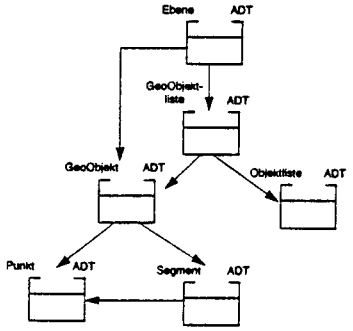

Abbildung 1.3: Transformation nach der 2. Alternative.

d) Wir ergänzen das OOA-Modell aus Abbildung 1 der Aufgabenstellung durch die Klasse Kreis, die wir als Unterklasse der abstrakten Klasse *GeoObjekt* einführen. Da ein Kreis aus einem Mittelpunkt besteht, fügen wir noch die Konstruktionsbeziehung zwischen den Klassen *Punkt* und *Kreis* ein. Ein Kreis hat genau einen Mittelpunkt, während ein Punkt höchstens ein Teilobjekt eines Kreises sein darf. Wir berücksichtigen diese Einschränkungen, indem wir die beiden Enden der Konstruktionsbeziehung mit Kardinalitäten 1 bzw. 0,1 versehen. Die Forderung, daß jeder Kreismittelpunkt kein Teilobjekt eines anderen Objekts der Ebene (z.B. Segment) sein darf, wird im OOA-Modell graphisch nicht zum Ausdruck gebracht, sondern kann nur als Invariante der Klasse *Punkt* notiert werden (Abbildung 1.4).

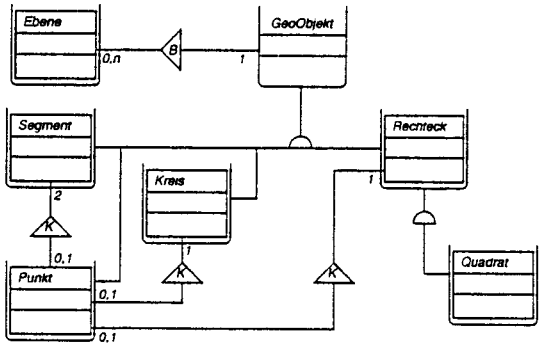

Abbildung 1.4: Ergänztes OOA-Modell.

Als nächstes setzen wir die Klasse *Rechteck* ins OOA-Modell ein. Ein Rechteck besitzt zwar einen Mittelpunkt und seine Ausdehnung wird durch Längenangaben charakterisiert, unterscheidet sich aber durch seine natürlichen Eigenschaften (z.B. Eckpunkte, Diagonalen usw.) so stark von einem Kreis, daß die Klasse *Rechteck* als Unterklasse der Klasse *Kreis* nicht in Frage kommt. Ansonsten erfolgt die Ergänzung des OOA-Modells im wesentlichen analog zur Ergänzung der Klasse *Kreis* (Abbildung 1.4).

Abschließend fügen wir die Klasse *Quadrat* als Unterklasse der Klasse *Rechteck* ein. Diese Vererbungsbeziehung finden wir auch in der "Problemwelt" wieder, in der Quadrate als spezialisierte Rechtecke charakterisiert wurden (Abbildung 1.4).

Aufgabe 2

Das Event-Response-System des Preview-Fensters muß auf Selektionen der Auswahlfelder "*Next*", "*Prev*", "*First*", "*Last*" und "*Quit*" und auf den Event InitMaxPage reagieren, der — wie in der Aufgabenstellung beschrieben — von der Applikationsschnittstelle an den Teildialog des Preview-Fensters verschickt wird.

```
TEILDIALOGBESCHREIBUNG Preview;
    EVENTS
        NextSelektiert,
        PrevSelektiert,
        FirstSelektiert,
        LastSelektiert,
        QuitSelektiert,
        InitMaxPage;
    FLAGS
        first, last;
    LOCALS
        @AktSeitenNr,
        @MaxSeitenNr : CARDINAL;
REGELN
    RULE INITIALLY DOES
        SEND FirstPage;
        @AktSeitenNr := 1;
        first := TRUE;
        last := TRUE;
        SEND CalculateMaxPage;
        SEND InitTitle;
        UpdateActDisplay.Number := @AktSeitenNr;
        SEND UpdateActDisplay;
        SEND UpdatePage
    END;
    RULE NextSelektiert CONDITION ¬ last DOES
        SEND NextPage;
        @AktSeitenNr := @AktSeitenNr+1;
        first := FALSE;
        last := (@MaxSeitenNr = @AktSeitenNr);
        UpdateActDisplay.Number := @AktSeitenNr;
        SEND UpdateActDisplay;
        SEND UpdatePage
    END;
    RULE PrevSelektiert CONDITION ¬ first DOES
        SEND PreviousPage;
        @AktSeitenNr := @AktSeitenNr-1;
        first := (1 = @AktSeitenNr);
        last := FALSE;
        UpdateActDisplay.Number := @AktSeitenNr;
        SEND UpdateActDisplay;
        SEND UpdatePage
    END;
    RULE FirstSelektiert CONDITION ¬ first DOES
        SEND FirstPage;
        @AktSeitenNr := 1;
        first := TRUE;
        last := FALSE;
        UpdateActDisplay.Number := @AktSeitenNr;
        SEND UpdateActDisplay;
        SEND UpdatePage
    END;
    RULE LastSelektiert CONDITION ¬ last DOES
        SEND LastPage;
        @AktSeitenNr := @MaxSeitenNr;
        first := FALSE;
        last := TRUE;
        UpdateActDisplay.Number := @AktSeitenNr;
        SEND UpdateActDisplay;
        SEND UpdatePage
    END;
    RULE QuitSelektiert DOES
        SEND Programmende
    END;
    RULE InitMaxPage DOES
        @MaxSeitenNr := InitMaxPage.Number;
        last := (@MaxSeitenNr = @AktSeitenNr);
        InitMaxDisplay.Number := @MaxSeitenNr;
        SEND InitMaxDisplay
    END;
END TEILDIALOGBESCHREIBUNG Preview;
```

Abbildung 2.1: Die Teildialogbeschreibung Preview-*Fensters.*

Aufgabe 3

Der in Aufgabe 2 vorgestellte Previewer besteht aus einem einzigen Fenster, das außerdem auch keine Teilfenster besitzt, die als eigenständige Szene modelliert werden könnten. Daher enthält das UIA-Modell des Previewers nur die nicht-multiple und nicht-modale Szene Preview (vgl. Abbildung 3.1). Die Benutzeraktionen der Szene Preview entsprechen den Auswahlfeldern und können direkt aus Abbildung 3 der Aufgabenstellung übernommen werden.

Die zentrale Aufgabe des Previewers ist die Darstellung einer bestimmten Seite des aktuellen Dokuments auf dem Bildschirm. Aus dieser Kurzbeschreibung können wir bereits Klassensichten und Klassenanker des UIA-Modells ableiten. Der Klassenanker actDocument enthält das bearbeitete Dokument und wird in der Szene Preview durch die Klassensicht DocumentView dargestellt. Der Klassenanker actPage enthält die auf dem Bildschirm dargestellte Dokument-

SZENE Preview

MULTIPEL	Nein
MODAL	Nein
GEGENSEITIGER AUSSCHLUSS	—
KONTROLLE ÜBER	—
ERZEUGUNG VON	—
NAVIGATION ZU	—

ANKER	LOKAL
actDocument	—
actPage	—

SICHTEN		ANKER
DocumentView	←	actDocument
PageView	←	actPage

BENUTZERAKTIONEN
- Next
- Prev
- First
- Last
- Quit

ERGÄNZENDE BESCHREIBUNGEN
—

Abbildung 3.1: Die Szene Preview.

seite und wird in der Szene Preview durch die Klassensicht PageView präsentiert.

Die Klassensicht DocumentView stellt den Titel eines *Document*-Objekts nicht-änderbar auf dem Bildschirm dar (Abbildung 3.2), während die Klassensicht PageView alle Attribute eines Objekts der Klasse *Page* nicht-änderbar auf dem Bildschirm anzeigt (Abbildung 3.3).

Im bisherigen UIA-Modell haben wir noch nicht festgelegt, welche Klassensichten die Informationen der Anzeigefeldern "*Act.*:" und "*Max.*:" auf dem Bildschirm präsentieren. Beide Anzeigefelder stellen Informationen dar, die nicht aus Attributen der Klassen *Document* oder *Page* abgeleitet werden können, sondern die mit Standarddiensten einer Behälterbeziehung erfragt werden können. Wir ergänzen daher die Klassensichten PageView und DocumentView jeweils durch eine Darstellung der Behälterbeziehung, welche die aktuelle Position des Objekts im Behälter bzw. die Anzahl der enthaltenen Objekte dieser Behälterbeziehung angeben.

SICHT DocumentView

KLASSE	Document
KAPAZITÄT	1

ATTRIBUTE	ÄNDERBAR
Title	Nein

OBJEKTBEZIEHUNGEN	KARDINALITÄT	ASSOZIERTE SICHTEN	ÄNDERBAR
Behälterbez. zur Klasse Page	1	—	Nein

ERGÄNZENDE BESCHREIBUNGEN
Die Darstellung der Behälterbeziehung zur Klasse Page erfolgt durch die Angabe der aktuellen Position und der Anzahl der enthaltenen Pageobjekte

Abbildung 3.2: Die Klassensicht DocumentView.

SICHT PageView

KLASSE	Page
KAPAZITÄT	1

ATTRIBUTE	ÄNDERBAR
Hight	Nein
Width	Nein
Content	Nein

OBJEKTBEZIEHUNGEN	KARDINALITÄT	ASSOZIERTE SICHTEN	ÄNDERBAR
Teil der Klasse Document	1	—	Nein

ERGÄNZENDE BESCHREIBUNGEN
Die Darstellung der Behälterbeziehung zur Klasse Document erfolgt durch die Angabe der aktuellen Position des darzustellenden Page-Objekts innerhalb seines Behälters

Abbildung 3.3: Die Klassensicht PageView.

Die Klassenanker actDocument und actPage referenzieren ein Objekt der Klasse *Document* bzw. *Page*, d.h. beide Anker haben die Kapazität 1. Die Kontrollbeziehung zwischen den Klassenankern spiegelt die Behälterbeziehung zwischen den Objekten beider Klassen wieder. Da das vorliegende UIA-Modell bisher nur eine nicht-multiple und nicht-modale Szene enthält, ist die Lokalität eines Klassenankers nicht erforderlich.

ANKER actDocument

KLASSE	Document
MULTIPEL	Nein
KAPAZITÄT	1
KONTROLLE ÜBER	actPage
LOKAL IN	—

ANKER actPage

KLASSE	Page
MULTIPEL	Nein
KAPAZITÄT	1
KONTROLLE ÜBER	—
LOKAL IN	—

Abbildung 3.4: Die Klassenanker actDocument *und* actPage.

Zum Abschluß ergänzen wir die Lösung zur Aufgabe 3 und stellen in Abbildung 3.5 die Präsentations- und Kontrollbeziehung des UIA-Modells des Previewers noch einmal graphisch dar.

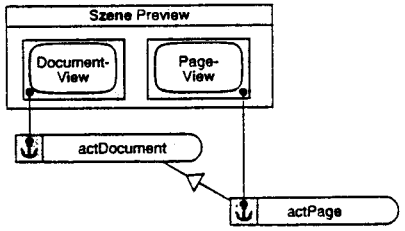

Abbildung 3.5: Previewer: Klassensichten und Präsentationsbeziehungen.

Aufgabe 4

a) Um festzustellen, welche der Prädikate A,B und C keinen Invarianten der `WHILE`-Schleife des Programms `ApproxRoot` sein können, bestimmen wir zunächst die Werte der Variablen `Z` und `N` für `X = 1` und `X = 2` vor jedem Schleifendurchlauf (vgl. Tabelle 4.1).

Iteration	*Variablenwerte*			
	X=1		X=2	
	Z	N	Z	N
vor dem 1. Durchlauf	1	1	1	1
vor dem 2. Durchlauf	–	–	3	2

Tabelle 4.1: Variablenbelegung der Prozedur `ApproxRoot`.

- *Prädikat* $A \equiv Z^2 < N^2 * (X+1) \ \wedge \ 1 \le Z$.
  Wir werden in Teilaufgabe (b) nachweisen, daß A ist eine Schleifeninvariante des Programms `ApproxRoot` ist.
- *Prädikat* $B \equiv Z^2 < N^2 * X \ \wedge \ 1 \le Z$
  kommt als Schleifeninvariante nicht in Frage, da für `X = 1` bereits vor dem ersten Durchlauf wegen `Z = 1` und `N = 1` die Bedingung $Z^2 = N^2 * X$ gilt. Allgemein ist das Prädikat B immer dann nicht erfüllt, wenn `X` eine ganzzahlige Wurzel besitzt.
- *Prädikat* $C \equiv 1 \le Z \le X \ \wedge \ N \le X$
  kann keine Schleifeninvariante sein, da für `X = 2` vor dem zweiten Durchlauf wegen `Z = 3` und `N = 2` die Bedingung `Z > X` gilt. Allgemein wird das Prädikat C zu `FALSE` ausgewertet, sobald im `ELSE`-Zweig von `ApproxRoot` die Anweisung `Z := Z*X+1` ausgeführt wurde.

b) Zur Abkürzung setzen wir

$$\begin{aligned} INV &\equiv Z^2 < N^2 * (X+1) \ \wedge \ 1 \le Z \\ B_{WHILE} &\equiv Z^2 \le N^2 * (X-1) \\ \text{sowie} \quad B_{IF} &\equiv (Z+1)^2 < N^2 * (X+1). \end{aligned}$$

Um die Gültigkeit der Programmformel $\{P_{11}\}\ S_{WHILE}\ \{P_{21}\}$ nachzuweisen, müssen wir die Prämisse der `WHILE`-Regel, also die Gültigkeit der Programmformel

```
      {P11 ≡ INV ∧ BWHILE}
13    IF (Z+1)*(Z+1) < N*N*(X+1) THEN
14       Z := Z+1
15    ELSE                                          (1)
16       Z := Z*X+1;
17       N := N*X
18    END
      {INV}
```

beweisen.

Wir untersuchen die beiden Alternativen der `IF`-Anweisung getrennt:

(i) `THEN`-Zweig:

Wir zeigen die Gültigkeit der Programmformel

```
      {INV ∧ BWHILE ∧ BIF}
14    Z := Z+1;                                     (2)
      {INV}.
```

Nun gilt mit dem Zuweisungsaxiom die Programmformel

```
      {(Z+1)² < N²*(X+1) ∧ 1 ≤ Z+1}
14    Z := Z+1
      {Z² < N²*(X+1) ∧ 1 ≤ Z}
```

Mit Hilfe der zweiten Teilbedingung $Z \ge 1$ der Invarianten INV folgt aus ihrer ersten Teilbedingung $0 < N^2 * (X+1)$ und damit die Aussage $N^2 \ge 1$. Aus $INV \wedge B_{WHILE} \wedge B_{IF}$ von (2) erhalten wir dann wegen $Z < Z+1$ und $N^2 * (X-1) < N^2 * (X+1)$ die logischen Implikationen

$$\begin{aligned} & INV \wedge B_{WHILE} \wedge B_{IF} \\ \Rightarrow \quad & 1 \le Z \ \wedge \ B_{IF} \\ \Rightarrow \quad & (Z+1)^2 < N^2 * (X+1) \ \wedge \ 1 \le Z+1. \end{aligned}$$

Aus der zweiten Konsequenzregel ergibt sich die Gültigkeit von (2).

(ii) `ELSE`-Zweig:

Um die Idee des formalen Beweises zu veranschaulichen, betrachten zunächst die Vorgehensweise des Programms `ApproxRoot` etwas näher. `ApproxRoot` approximiert die Wurzel von `X` durch eine streng monoton steigende Folge `Z/N`. Der Zähler `Z` wird nur dann um `1` vergrößert, wenn $(Z+1)^2/N^2$ unterhalb der oberen Schranke `X+1` liegt. Andernfalls werden im `ELSE`-Zweig Zähler und Nenner durch `X` erweitert und der Zähler um `1` vergrößert. Da die Erweiterung von Zähler und Nenner für die Invariante INV unkritisch ist, müssen wir im `ELSE`-Zweig insbesondere zeigen, daß unter der Voraussetzung von INV, B_{WHILE} und $\neg B_{IF}$ die durchgeführte Vergrößerung des Zählers die Invarianteneigenschaft nicht zerstört.

Formal müssen die Gültigkeit der Programmformel

```
      {INV ∧ BWHILE ∧ ¬BIF}
16    Z := Z*X+1;
17    N := N*X                                      (3)
      {INV}
```

zeigen. Dafür wenden wir das Zuweisungsaxiom und die Sequenzregel an und erhalten die Gültigkeit der Programmformel

```
      {(Z*X+1)² < (N*X)²*(X+1) ∧ 1 ≤ Z*X+1}
16    Z := Z*X+1;
      {Z² < (N*X)²*(X+1) ∧ 1 ≤ Z}
17    N := N*X
      {Z² < N²*(X+1) ∧ 1 ≤ Z}
```

Wir setzen zur Abkürzung

$$P_{TEMP} \equiv (Z*X+1)^2 < (N*X)^2 * (X+1) \ \wedge \ 1 \le Z*X+1.$$

Um die Gültigkeit der Programmformel (3) mit Hilfe der zweiten Konsequenzregel zu zeigen, müssen wir die Implikation $INV \wedge B_{WHILE} \wedge \neg B_{IF} \Rightarrow P_{TEMP}$ beweisen.

Dazu leiten wir aus der Voraussetzung zunächst zwei Hilfsaussagen über X und N ab:

- Aus INV und B_{WHILE} folgt zunächst die Bedingung $X > 1$.
- B_{WHILE} ist äquivalent zu $Z^2+2N^2 \leq N^2*(X+1)$, d.h. zusammen mit $\neg B_{IF}$ gilt:

$$\begin{array}{ll} & Z^2+2N^2 \leq N^2*(X+1) \leq (Z+1)^2 \\ \Leftrightarrow & Z^2+2N^2 \leq N^2*(X+1) \leq Z^2+2Z+1 \\ \Rightarrow & 2N^2 \leq 2Z+1. \end{array}$$

Mit Hilfe dieser zwei Hilfsaussagen können wir nun folgern

$$\begin{array}{lll} & INV \wedge B_{WHILE} \wedge \neg B_{IF} & \\ \Leftrightarrow & Z^2 < N^2*(X+1) \wedge 1 \leq Z \wedge Z^2 \leq N^2*(X-1) \wedge (Z+1)^2 \geq N^2*(X+1) & \\ \Rightarrow & Z^2 \leq N^2*(X-1) \wedge 1 \leq Z & \mid \text{wegen } X > 1 \\ \Rightarrow & Z^2+2N^2 \leq N^2*(X+1) \wedge 1 \leq Z*X+1 & \\ \Rightarrow & X^2Z^2+2N^2X^2 \leq X^2N^2*(X+1) \wedge 1 \leq Z*X+1 & \mid \text{wegen } 2N^2 \leq 2Z+1 \text{ und } X > 1 \\ \Rightarrow & X^2Z^2+2ZX+1 < X^2N^2*(X+1) \wedge 1 \leq Z*X+1 & \\ \Leftrightarrow & (Z*X+1)^2 < (N*X)^2*(X+1) \wedge 1 \leq Z*X+1 & \\ \Leftrightarrow & P_{TEMP} & \end{array}$$

Es gilt also $INV \wedge B_{WHILE} \wedge \neg B_{IF} \Rightarrow P_{TEMP}$ und damit ist nach der zweiten Konsequenzregel auch die Programmformel (3) gültig.

Aus (i) und (ii) ergibt sich mit Hilfe der ersten Bedingungsregel die Gültigkeit von Programmformel (1) und damit folgt durch Anwendung der WHILE-Regel die Gültigkeit der Programmformel $\{P_{11}\}\ S_{WHILE}\ \{P_{21}\}$.

c) Zum Nachweis der partiellen Korrektheit des Programms ApproxRoot müssen wir die Gültigkeit der Programmformel von $\{P\}\ S_{ApproxRoot}\ \{Q\}$ zeigen. Da wir in Teilaufgabe (b) die Gültigkeit der Programmformel $\{INV\}\ S_{WHILE}\ \{INV \wedge \neg B_{WHILE}\}$ bereits nachgewiesen haben, genügt zum Beweis der partiellen Korrektheit der Nachweis von:

(i) $\{P\}\ N:=1;\ Z:=1\ \{INV\}$

und (ii) $INV \wedge \neg B_{WHILE} \Rightarrow Q$.

zu (i): Unter Verwendung des Zuweisungsaxioms und der Sequenzregel folgt:

```
        {1<(X+1) ∧ 1 ≤ 1}
   9    N := 1;
  10    Z := 1
        {Z² < N²*(X+1) ∧ 1 ≤ Z}
```

Da $P \equiv X \geq 1$ folgt mit Hilfe der zweiten Konsequenzregel die Behauptung (i).

zu (ii): Mit $N^2 \geq 1$ (vgl. Teilaufgabe (b), ELSE-Zweig) können wir die folgenden Äquivalenzen bzw. Implikationen direkt ableiten:

$$\begin{array}{lll} & INV \wedge \neg B_{WHILE} & \\ \Leftrightarrow & Z^2 < N^2*(X+1) \wedge 1 \leq Z \wedge Z^2 > N^2*(X-1) & \\ \Rightarrow & N^2*(X-1) < Z^2 < N^2*(X+1) & \mid N^2 \geq 1 \\ \Leftrightarrow & -1 < Z^2/N^2-X < 1 & \\ \Rightarrow & Q & \end{array}$$

Aus (i) und (ii) folgt unter Berücksichtigung der ersten und zweiten Konsequenzregel folgt die Gültigkeit der Programmformel von $\{P\}\ S_{ApproxRoot}\ \{Q\}$, also die partielle Korrektheit des Programms ApproxRoot.

d) Wir zeigen die totale Korrektheit des Programms ApproxRoot mit Hilfe einer Terminierungsfunktion, die wir aus der Invarianten INV der WHILE-Schleife wie folgt ableiten:

$$\begin{array}{lll} & INV \equiv Z^2 < N^2*(X+1) \wedge 1 \leq Z & \\ \Rightarrow & Z^2 < N^2*(X+1) & \mid \text{wegen } N \leq X \text{ (lt. Voraussetzung)} \\ \Rightarrow & Z^2 < X^2*(X+1) & \\ \Leftrightarrow & 0 < X^3+X^2-Z^2 & \end{array} \qquad (4)$$

Wir definieren die Abbildung τ als:

$$\tau: \mathbb{Z}^3 \rightarrow \mathbb{Z} \text{ mit } \tau(X, Z, N) = X^3+X^2-Z^2$$

Um den Nachweis zu erbringen, daß die Abbildung τ tatsächlich eine Terminierungsfunktion ist, müssen wir zeigen, daß τ die Eigenschaften (w1) - (w3) erfüllt.

zu (w1): Die Eigenschaft (w1) ist laut Definition der Abbildung τ erfüllt.

zu (w2): Wir müssen zeigen, daß unter der Voraussetzung von $INV \wedge B_{WHILE}$ jede Ausführung von S_{WHILE} den Funktionswert von τ dekrementiert. Dazu müssen wir wieder zwischen zwei Fällen unterscheiden:

i) Durchläuft die WHILE-Schleife in einem Iterationsschritt den THEN-Zweig, wird der Wert von Z um 1 erhöht. Die Schleifeninvariante INV besagt, daß Z stets größer als 0 ist, daher folgt $X^3+X^2-Z^2 > X^3+X^2-(Z+1)^2$ und somit

$$\{\tau(X, Z, N) = t\}\ Z:=Z+1\ \{\tau(X, Z, N) < t\}$$

ii) Durchläuft die WHILE-Schleife in einem Iterationsschritt den ELSE-Zweig, wird der Wert von Z mit X multipliziert und anschließend um 1 erhöht. Da aber wegen $Z > 0$ auch $X^3+X^2-Z^2 > X^3+X^2-(2Z+1)^2$ gilt, folgt die Aussage:

$$\{\tau(X, Z, N) = t\}\ Z:=Z*X+1;\ N:=N*X\ \{\tau(X, Z, N) < t\}$$

Aus (i) und (ii) folgt zusammen: $\{\tau(X, Z, N) = t\}\ S_{WHILE}\ \{\tau(X, Z, N) < t\}$; d.h. die Abbildung τ ist streng monoton fallend.

zu (w3): Unter der Voraussetzung von $INV \wedge B_{WHILE}$ folgt (s.o., Herleitung von τ):

$$\tau(X, Z, N) > 0.$$