(**5 Punkte**) Gegeben ist der folgende Programmausschnitt mit der unten stehenden Nachbedingung `x > 24`. Geben Sie die schwächste Vorbedingung an und die Reihenfolge, in der Sie die einzelnen Zusicherungen berechnen.

```
  {                                                              }

x := 2*x;

  {                                                              }

  {                                                              }

IF x < 10 THEN

  {                                                              }

    y := 3*x - 4;

  {                                                              }
```

```
  {                                                          }

    y := 3*x - 4;

  {                                                          }

ELSE

  {                                                          }

    y := x - 12;

  {                                                          }

END IF;

  {                                                          }

x := 2*y + 8;

  { x > 24                                                   }
```

(**5 Punkte**) Gegeben ist der folgende Programmausschnitt mit der unten stehenden Nachbedingung `y > 0`. Leiten Sie eine notwendige Zusicherung her, die zu Beginn des Programms gelten muss, damit am Ende die Nachbedingung `y > 0` zutrifft. Vereinfachen Sie die Zusicherungen, wenn es möglich ist.

```
  {                                                         }
x := 4 - y ;
  {                                                         }
  {                                                         }
IF x <= 4 THEN
  {                                                         }
    x := 9 - x;
  {                                                         }
ELSE
  {                                                         }
    x := 2 * x - 1;
  {                                                         }
END IF;
  {                                                         }
y := x - 5;
  { y > 0                                                   }
```

v

Gegeben ist die folgende `while`-Schleife (alle Variablen sind vom Typ `integer`).

```
WHILE i > 0 LOOP
    z := z + i;
    i := i - 1;
END LOOP;
```

a) (**3 Punkte**) Angenommen, die Zusicherung *Inv* sei eine Invariante für obige `while`-Schleife. Beschreiben Sie die Hoaresche Regel für die `while`-Schleife: Was müssen Sie bei der Anwendung dieser Regel nachweisen und welche Folgerung können Sie dann schließen?

b) (**2+2+2 Punkte**) Welche der folgenden Zuweisungen sind Invarianten für obige `while`-Schleife? Begründen Sie Ihre Antwort.

- `{ z = i * (i+1) / 2 }`
- `{ i * (i+1) / 2 + z = 42 }`
- `{ i ≥ 0 }`

Verifikation: Gegeben ist das folgende Programm.

a) (**1 Punkt**) Geben Sie die hierdurch berechnete Funktion `f` an:

`f(a,b)=`________________________ für alle `a,b` $\in \mathbb{N}_0$

b) (**6 Punkte**) und beweisen Sie mit den Hoareschen Regeln, dass diese Funktion tatsächlich berechnet wird.

```
PROCEDURE unbekannt IS
x, y, z: NATURAL := 0;

      {                                                         }

BEGIN Get (x); Get (y);

      {                                                         }

WHILE y > 0 LOOP

      {                                                         }

    z:=z+x;

      {                                                         }

    y:=y-1;

      {                                                         }

END LOOP;

      {                                                         }

      {      z =                                                }

Put(z);
END;
```