1) a) Wir nehmen an, dass ein $m > 0$ und Indizes existieren, sodass $x_{i_1} x_{i_2} \dots x_{i_m} = y_{i_1} y_{i_2} \dots y_{i_m}$. Wir unterscheiden folgende Fälle:

i. Sei $i_1 = 1$, dann gilt für alle $m > 0$, dass $|x_{i_1} x_{i_2} \dots x_{i_m}| \neq |y_{i_1} y_{i_2} \dots y_{i_m}|$, d.h $x \neq y$.

ii. Sei $i_1 = 2$, dann gilt $101 x_{i_2} \dots x_{i_m} \neq 100 y_{i_2} \dots y_{i_m}$.

iii. Sei $i_1 = 3$, dann gilt $11000 x_{i_2} \dots x_{i_m} \neq 101 y_{i_2} \dots y_{i_m}$.

Unabhängig von der Wahl für i_1 stimmen die Zeichenfolgen nicht überein. Daher ist unsere Annahme falsch und es existiert keine Lösung.

b) $m = 3$ mit Indizes $(2, 1, 3)$

2) *Lösung.* Die TM $M = (\{1, 2, 3, t, r\}, \{\mathsf{a}, \mathsf{b}\}, \{\mathsf{a}, \mathsf{b}, \vdash, \sqcup\}, \vdash, \sqcup, \delta, 1, t, r)$, wobei δ wie folgt definiert ist:

δ	$\vdash$	a	b	$\sqcup$
1	$(1, \vdash, R)$	$(2, \mathsf{a}, R)$	$(1, \mathsf{b}, R)$	$(r, \sqcup, R)$
2	–	$(2, \mathsf{a}, R)$	$(3, \mathsf{b}, R)$	$(r, \sqcup, R)$
3	–	(t, a, R)	$(1, \mathsf{b}, R)$	$(r, \sqcup, R)$
t	–	(t, a, R)	(t, b, R)	$(t, \sqcup, R)$
r	–	(r, a, R)	(r, b, R)	$(r, \sqcup, R)$

$$(1, \vdash \sqcup\sqcup^\infty, 0) \underset{M}{\longrightarrow} (1, \vdash \sqcup\sqcup^\infty, 1) \underset{M}{\longrightarrow} (r, \vdash \sqcup\sqcup^\infty, 2)$$

$$\begin{aligned}
&(1, \vdash \mathsf{babba}\sqcup^\infty, 0) \underset{M}{\longrightarrow} \\
&(1, \vdash \mathsf{babba}\sqcup^\infty, 1) \underset{M}{\longrightarrow} \\
&(1, \vdash \mathsf{babba}\sqcup^\infty, 2) \underset{M}{\longrightarrow} \\
&(2, \vdash \mathsf{babba}\sqcup^\infty, 3) \underset{M}{\longrightarrow} \\
&(r, \vdash \mathsf{babba}\sqcup^\infty, 4)
\end{aligned}$$

$$\begin{aligned}
&(1, \vdash \mathsf{abab}\sqcup^\infty, 0) \underset{M}{\longrightarrow} \\
&(1, \vdash \mathsf{abab}\sqcup^\infty, 1) \underset{M}{\longrightarrow} \\
&(2, \vdash \mathsf{abab}\sqcup^\infty, 2) \underset{M}{\longrightarrow} \\
&(3, \vdash \mathsf{abab}\sqcup^\infty, 3) \underset{M}{\longrightarrow} \\
&(t, \vdash \mathsf{abab}\sqcup^\infty, 4) \underset{M}{\longrightarrow} \\
&(t, \vdash \mathsf{abab}\sqcup^\infty, 5)
\end{aligned}$$

□

3) *Lösung.* Wie Sie bereits aus der VO Rechnerarchitektur erfahren haben ist der Hauptbestandteil einer CPU die ALU und die Register. D.h. Programme sind (sehr vereinfacht ausgedrückt) Sequenzen von arithmetischen Operationen und Register die erlauben Werte zu schreiben und zu lesen (speichern). Das Registriermaschinenmodell ist eine Möglichkeit diesen essenziellen Teil darzustellen und darüber Aussagen zu treffen.

Das folgende Programm führt die ganzzahlige Division von x_1/x_2 durch. Das Ergbnis der Division steht nach Ausführung in Register x_3, der Rest der Division in Register x_4.

```
while x1 ≠ 0 do
  while x2 ≠ 0 do
    x2 := x2 − 1;
    x1 := x1 − 1;
    x4 := x4 + 1;
    P_null(x1, x5, x6);

    # falls x1 gleich 0 ist wollen wir die Schleife beenden
    while x5 ≠ 0 do

      # setze x5 zurueck auf 0 sodass der folgende Code genau einmal
      # ausgefuehrt wird
      x5 := x5 − 1;

      # falls  x2 gleichzeitig 0 wurde muessen wir den Rest auf 0 setzen
      P_null(x2, x5, x6);
      while x5 ≠ 0 do
        x5 := x5 − 1;
        while x4 ≠ 0 do
          x4 := x4 − 1
        end
      end;

      # setze x2 auf 0 um die Schleife zu beenden
      while x2 ≠ 0 do
        x2 := x2 − 1
      end
    end

  end;

  # Fuer den Fall, dass x1 noch nicht 0 ist, schreiben wir den Wert
```

```
  # aus Register x4 wieder in Register x2,
  # um ihn erneut von x1 abzuziehen.
  P_notnull(x1, x5, x6);
  while x5 ≠ 0 do
    x5 := x5 − 1;
    while x4 ≠ 0 do
      x4 := x4 − 1;
      x2 := x2 + 1
    end
  end;

  # x3 wird nur nach oben gezaehlt falls Register x4 (Rest) 0 ist.
  # Also wenn x1 noch nicht 0 ist oder wenn x1 und x2 gleichzeitig
  # 0 wurden.)
  P_null(x4, x5, x6);
  while x5 ≠ 0 do
    x5 := x5 − 1;
    x3 := x3 + 1
  end

end
```

Das Hilfsprogramm $P_{null}(x_1, x_2, x_3)$ testet ob das Register $x_1 = 0$ ist. In diesem Fall wird x_2 auf 1 gesetzt, ansonsten auf 0. Register x_3 ist ein Hilfsregister, um sicherzustellen, dass x_1 auch nach der Ausführung noch denselben Wert hat wie zu Beginn.

```
while x3 ≠ 0 do
  x3 := x3 − 1 # setze x3 auf 0
end;
while x2 ≠ 0 do
  x2 := x2 − 1 # setze x2 auf 0
end;
x2 := x2 + 1; # setze x2 auf 1
while x1 ≠ 0 do
  x2 := x2 − 1;
  x1 := x1 − 1;
  x3 := x3 + 1 # speichere den Wert aus Register x1 in x3
end;
# schreibe den gespeicherten Wert in x3 zurueck nach x1
while x3 ≠ 0 do
  x3 := x3 − 1;
  x1 := x1 + 1
end
```

Das Hilfsprogramm $P_{notnull}(x_1, x_2, x_3)$ testet ob das Register x_1 ungleich 0 ist. In diesem Fall wird x_2 auf 1 gesetzt, ansonsten auf 0. Register x_3 ist ein Hilfsregister, um sicherzustellen, dass x_1 auch nach der Ausführung noch denselben Wert hat wie zu Beginn.

```
while x3 ≠ 0 do
  x3 := x3 - 1 # setze x3 auf 0
end;
while x2 ≠ 0 do
  x2 := x2 - 1 # setze x2 auf 0
end;
while x1 ≠ 0 do
  x2 := x2 + 1;

  # durch diese Schleife wird verhindert,
  # dass der Wert von x2 groesser als 1 wird
  while x1 ≠ 0 do
    x1 := x1 - 1;
    x3 := x3 + 1 # speichere den Wert aus Register x1 in x3
  end
end;
# schreibe den gespeicherten Wert in x3 zurueck nach x1
while x3 ≠ 0 do
  x3 := x3 - 1;
  x1 := x1 + 1
end
```

□