1. (3 Punkte) Für welche ganzen Zahlen `n` halten die drei Funktionen an?

```
def f(n):
    while n != 0:
        n = n - 2

def g(n):
    while n > 0:
        n = n - 3

def h(n):
    while n != 1:
        if n % 2 == 0:
            n = n // 2
        else:
            n = n + 1
```

Lösung: `f` hält für alle geraden `n` ≥ 0. Bei ungeraden Zahlen wird die 0 übersprungen, negative Zahlen werden immer kleiner.

`g` hält für alle `n`. Für `n` ≤ 0 wird die Schleife gar nicht betreten.

`h` hält für alle `n` ≥ 1. Bei 0 bleibt `n` immer 0. Negative Zahlen kommen über -1 zur 0 und bleiben dort.

2. (2 Punkte) Die Funktion `collatz` aus dem Notebook halbiert eine gerade Zahl, eine ungerade Zahl wird mit 3 multipliziert und um 1 erhöht, bis die 1 erreicht ist.
   (a) Schreibe die Folge für `n = 7` auf. Wie viele Schritte sind es?
   (b) Ob die Funktion für jede natürliche Zahl anhält, weiß niemand. Was müsste man finden, um die Vermutung zu widerlegen?

Lösung:

(a) 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, also 16 Schritte.

(b) Eine Startzahl, deren Folge nie die 1 erreicht: Entweder werden die Zahlen immer größer, oder die Folge kommt in einen Kreis, der die 1 nicht enthält.

3. (4 Punkte) Wir nehmen an, es gäbe eine Funktion `haltetest(pgm, inp)`, die `True` zurückgibt, wenn `pgm` angesetzt auf `inp` anhält, sonst `False`.

```
def quer(x):
    if haltetest(x, x):
        while True:
            pass
```

(a) Beschreibe in eigenen Worten, was `quer` macht.

(b) Zeige, dass die Frage, ob `quer(quer)` anhält, zu einem Widerspruch führt.

(c) Was folgt aus dem Widerspruch?

Lösung:

(a) `quer` fragt, ob das Programm `x` angesetzt auf sich selbst anhält. Wenn ja, läuft `quer` endlos, wenn nein, ist `quer` sofort fertig. `quer` macht also das Gegenteil von `x` mit `x`.

(b) Fall 1: `quer(quer)` hält an. Dann ist `haltetest(quer, quer) True`, also läuft `quer(quer)` in die Endlosschleife und hält nicht an. Widerspruch.

Fall 2: `quer(quer)` hält nicht an. Dann ist `haltetest(quer, quer) False`, also ist `quer(quer)` sofort fertig und hält an. Widerspruch.

(c) `quer` ist ein ganz gewöhnliches Programm. Falsch ist die Annahme: Es gibt keine Funktion `haltetest`. Das Halteproblem ist unentscheidbar.

4. (3 Punkte) (a) Jemand ersetzt im Beweis die Funktion `quer` durch

```
def gleich(x):
    if not haltetest(x, x):
        while True:
            pass
```

Führt die Frage, ob `gleich(gleich)` anhält, ebenfalls zu einem Widerspruch?

(b) Jemand meint: Man muss ein Programm nur eine Stunde laufen lassen. Ist es dann noch nicht fertig, hält es nicht an. Was ist falsch daran?

Lösung:

(a) Nein. Hält `gleich(gleich)` an, dann ist `haltetest(gleich, gleich) True`, die Bedingung ist falsch und `gleich(gleich)` hält an. Das passt. Hält `gleich(gleich)` nicht an, dann ist `haltetest False`, `gleich` läuft endlos. Das passt ebenfalls. `gleich` macht dasselbe wie `x` mit `x`, nicht das Gegenteil, deshalb entsteht kein Widerspruch.

(b) Ein Programm kann auch nach mehr als einer Stunde noch anhalten. Für jede feste Zeitgrenze gibt es Programme, die länger brauchen und dann doch fertig werden, z.B. eine Schleife, die eine Milliarde Mal mehr läuft. Die Antwort wäre bei diesen Programmen falsch.

5. (4 Punkte) Angenommen, es gäbe eine Funktion `druckthallo(pgm, inp)`. Sie gibt `True` zurück, wenn `pgm` angesetzt auf `inp` irgendwann `Hallo` ausgibt, sonst `False`.
   (a) Schreibe mit Hilfe von `druckthallo` eine Funktion `haltetest(pgm, inp)`. Du kannst annehmen, dass `pgm` selbst nichts ausgibt.
   Tipp: Schreibe innerhalb von `haltetest` eine neue Funktion, die zuerst `pgm(inp)` aufruft.
   (b) Was folgt daraus für `druckthallo`?

Lösung:

(a)
```
def haltetest(pgm, inp):
    def test(x):
        pgm(inp)
        print('Hallo')
    return druckthallo(test, inp)
```

`test` gibt genau dann `Hallo` aus, wenn `pgm(inp)` anhält. Also liefert `druckthallo(test, inp)` genau die Antwort auf das Halteproblem.

(b) Es kann `druckthallo` nicht geben, sonst gäbe es auch `haltetest`. Die Frage, ob ein Programm `Hallo` ausgibt, ist also auch unentscheidbar.

6. (2 Punkte) Gegeben sind die Dominosteine `1: (1, 111)`, `2: (10111, 10)` und `3: (10, 0)`, jeweils (oben, unten). Schreibe für beide Folgen das obere und das untere Gesamtwort auf. Ist die Folge eine Lösung?
   (a) 2, 1, 1, 3
   (b) 2, 3, 1

Lösung:

(a) oben `101111110`, unten `101111110`, eine Lösung.

(b) oben `10111101`, unten `100111`, keine Lösung.

7. (3 Punkte) Gegeben sind die Dominosteine `1: (0, 1)`, `2: (10, 0)` und `3: (0, 010)`, jeweils (oben, unten).
   (a) Mit welchem Stein muss eine Lösung beginnen? Begründe.
   (b) Finde eine Lösung mit vier Steinen. Gib nach jedem Stein den Überhang an.

Lösung:

(a) Mit Stein 3. Bei Stein 1 beginnen die Wörter mit 0 und 1, bei Stein 2 mit 1 und 0. Nur bei Stein 3 ist das obere Wort ein Anfang des unteren.

(b) Folge 3, 2, 1, 2:
nach 3: oben `0`, unten `010`, Überhang `10` unten
nach 2: oben `010`, unten `0100`, Überhang `0` unten
nach 1: oben `0100`, unten `01001`, Überhang `1` unten
nach 2: oben `010010`, unten `010010`, kein Überhang

8. (3 Punkte) Begründe, warum es für diese Dominosteine keine Lösung gibt. Die Steine sind jeweils als (oben, unten) angegeben.
   (a) `1: (011, 01)`, `2: (10, 1)`
   (b) `1: (01, 1)`, `2: (10, 0)`
   (c) `1: (0, 01)`, `2: (100, 1)`

Lösung:

(a) Bei jedem Stein ist das obere Wort ein Zeichen länger als das untere. Bei k Steinen ist oben also immer k Zeichen mehr, die Gesamtwörter sind nie gleich lang.

(b) Kein Stein passt am Anfang: Bei Stein 1 beginnt oben mit 0 und unten mit 1, bei Stein 2 ist es umgekehrt.

(c) Bei beiden Steinen endet das obere Wort mit 0 und das untere mit 1. Das obere Gesamtwort endet also immer mit 0, das untere mit 1.

9. (3 Punkte) (a) Beschreibe ein Verfahren, das für beliebige Dominosteine eine Lösung des PKP findet, wenn es eine gibt.
   (b) Warum entscheidet dieses Verfahren das PKP nicht?
   (c) Warum ist das Halteproblem semi-entscheidbar?

Lösung:

(a) Man probiert alle Folgen der Länge 1, dann alle der Länge 2, dann 3 usw. Für jede Länge gibt es nur endlich viele Folgen. Gibt es eine Lösung, dann kommt sie bei ihrer Länge an die Reihe, und das Verfahren antwortet Ja. (Geschickter: Breitensuche über die Überhänge.)

(b) Gibt es keine Lösung, dann läuft das Verfahren endlos und gibt nie die Antwort Nein. Ein Entscheidungsverfahren muss aber für jede Eingabe anhalten.

(c) Man lässt das Programm mit der Eingabe laufen. Hält es an, antwortet man Ja. Hält es nicht an, läuft auch dieses Verfahren endlos. Bei Ja-Antworten hält es also sicher an, bei Nein-Antworten vielleicht nie.

10. (4 Punkte) Hilberts Hotel hat die Zimmer 1, 2, 3, ..., alle sind belegt.
    (a) Drei neue Gäste kommen. In welches Zimmer zieht der Gast aus Zimmer n, welche Zimmer bekommen die neuen Gäste?
    (b) Ein Bus mit unendlich vielen Gästen kommt. Der Gast aus Zimmer n zieht in Zimmer $2n$, der Gast mit Platz k im Bus bekommt Zimmer $2k-1$. Welches Zimmer bekommt der Gast mit Platz 10 im Bus? Wohin zieht der Gast aus Zimmer 7?
    (c) Unendlich viele Busse kommen. Die alten Gäste sind Bus 0, ihr Platz ist die alte Zimmernummer. Der Gast mit Platz k in Bus b bekommt mit $s = b + k$ das Zimmer

$$\frac{(s-1)\cdot s}{2} + b + 1$$

    Welches Zimmer bekommt der Gast mit Platz 3 in Bus 2? Wohin zieht der alte Gast aus Zimmer 5? Wer bekommt Zimmer 12?

Lösung:

(a) Der Gast aus Zimmer n zieht in Zimmer $n+3$. Die neuen Gäste bekommen die Zimmer 1, 2 und 3.

(b) Platz 10 im Bus: Zimmer 19. Der Gast aus Zimmer 7 zieht in Zimmer 14.

(c) Platz 3 in Bus 2: $s = 5$, Zimmer $10 + 2 + 1 = 13$.
Alter Gast aus Zimmer 5: Bus 0, Platz 5, $s = 5$, Zimmer $10 + 0 + 1 = 11$.
Zimmer 12: Platz 4 in Bus 1, denn $s = 5$ und $10 + 1 + 1 = 12$.

11. (3 Punkte) Die natürlichen Zahlen werden so den ganzen Zahlen zugeordnet:

$$1 \mapsto 0, \quad 2 \mapsto 1, \quad 3 \mapsto -1, \quad 4 \mapsto 2, \quad 5 \mapsto -2, \quad \ldots$$

    (a) Welche ganzen Zahlen gehören zu 20 und zu 15?
    (b) Welche natürlichen Zahlen gehören zu 12 und zu -12?
    (c) Gib die Zuordnung als Formel an, getrennt für gerade und ungerade n.

Lösung:

(a) $20 \mapsto 10$, $15 \mapsto -7$

(b) $24 \mapsto 12$, $25 \mapsto -12$

(c) Für gerade n: $n \mapsto \frac{n}{2}$. Für ungerade n: $n \mapsto -\frac{n-1}{2}$.

12. (3 Punkte) Die positiven Brüche werden nach Cantor schräg abgezählt, kürzbare Brüche werden übersprungen. Die ersten elf sind

$$\frac{1}{1}, \frac{1}{2}, \frac{2}{1}, \frac{3}{1}, \frac{1}{3}, \frac{1}{4}, \frac{2}{3}, \frac{3}{2}, \frac{4}{1}, \frac{5}{1}, \frac{1}{5}$$

(a) Gib die Brüche mit den Nummern 12 bis 18 an.

(b) Welche Nummer hat $\frac{3}{5}$?

(c) Warum dürfen kürzbare Brüche übersprungen werden, warum müssen sie es sogar?

Lösung:

(a) $\frac{1}{6}$, $\frac{2}{5}$, $\frac{3}{4}$, $\frac{4}{3}$, $\frac{5}{2}$, $\frac{6}{1}$, $\frac{7}{1}$

(b) Nummer 20. Die Schräge mit $s = 8$ beginnt mit $\frac{7}{1}$ (18), dann $\frac{6}{2}$ (übersprungen), $\frac{5}{3}$ (19), $\frac{4}{4}$ (übersprungen), $\frac{3}{5}$ (20).

(c) Ein kürzbarer Bruch ist dieselbe Zahl wie ein gekürzter, der schon vorher drankam, z.B. $\frac{2}{2} = \frac{1}{1}$. Ohne Überspringen bekäme dieselbe Zahl mehrere Nummern, die Zuordnung wäre nicht 1:1.

13. (3 Punkte) Gegeben sind die Anfänge von sechs unendlichen 0-1-Folgen:

`110100...  001011...  101010...  111000...  000111...  010101...`

(a) Bilde die ersten sechs Stellen der Diagonalfolge d: An der Stelle i steht das Gegenteil der i-ten Stelle der i-ten Folge.

(b) Warum kommt d in keiner Liste von Folgen vor, die man so behandelt?

(c) Jemand schlägt vor, d einfach vorne in die Liste einzufügen. Warum ist die Liste dann immer noch nicht vollständig?

Lösung:

(a) Die Diagonale ist 1, 0, 1, 0, 1, 1, also $d =$ `010100...`

(b) d unterscheidet sich von der i-ten Folge an der Stelle i, für jedes i.

(c) Für die neue Liste kann man wieder eine Diagonalfolge bilden, die in ihr fehlt. Das klappt bei jeder Liste.

14. (2 Punkte) Angenommen, eine Liste aller Zahlen zwischen 0 und 1 beginnt mit

0,123456... 0,505050... 0,987654... 0,555555... 0,314159... 0,999999...

Bilde die ersten sechs Nachkommastellen der Zahl x aus dem Diagonalverfahren: Ist die i-te Nachkommastelle der i-ten Zahl eine 5, dann ist die i-te Stelle von x eine 4, sonst eine 5.

Lösung: Die Diagonale ist 1, 0, 7, 5, 5, 9, also $x = 0{,}555445...$

15. (4 Punkte) Richtig oder falsch? Begründe jeweils kurz.
    (a) Das Halteproblem ist unentscheidbar, weil die Rechner zu langsam sind.
    (b) Für ein bestimmtes Programm mit einer bestimmten Eingabe kann man oft sicher sagen, ob es anhält.
    (c) $\mathbb{N}$ ist eine echte Teilmenge von $\mathbb{Z}$, also ist $\mathbb{Z}$ mächtiger als $\mathbb{N}$.
    (d) $\mathbb{Q}$ ist überabzählbar, weil zwischen zwei Brüchen immer noch ein weiterer Bruch liegt.
    (e) Es gibt nur abzählbar viele Python-Programme.
    (f) Für fast alle Entscheidungsprobleme gibt es keinen Algorithmus.

Lösung:

(a) Falsch. Es gibt grundsätzlich kein Programm, das die Frage für alle Programme und Eingaben richtig beantwortet, egal wie schnell der Rechner ist.

(b) Richtig. Unentscheidbar heißt nur, dass kein Algorithmus alle Fälle richtig beantwortet. Viele einzelne Fälle sind leicht, z.B. ein Programm ohne Schleifen und ohne Rekursion hält immer.

(c) Falsch. Die Zuordnung $1 \mapsto 0, 2 \mapsto 1, 3 \mapsto -1, \ldots$ ist 1:1, die Mengen sind gleichmächtig.

(d) Falsch. Mit der Abzählung von Cantor bekommt jeder Bruch eine Nummer, $\mathbb{Q}$ ist abzählbar.

(e) Richtig. Jedes Programm ist ein endlicher Text, also eine Folge von Bytes und damit eine natürliche Zahl.

(f) Richtig. Es gibt überabzählbar viele Entscheidungsprobleme (so viele wie 0-1-Folgen), aber nur abzählbar viele Programme. Jedes Programm entscheidet höchstens ein Problem.