Prof. Dr. Visvanathan Ramesh
Alperen Kantarcı
Gamze Akyol

Goethe-Universität Frankfurt am Main
Institut für Informatik

Programmierparadigmen und Compilerbau

Sommersemester 2023

Übungsblatt Nr. 4

Abgabe Mittwoch 22.06.2023, 10:00 in Moodle!

Aufgabe 0 **(0 Punkte)**

Lesen Sie die folgenden Tipps!

- Gruppenarbeit ist **nicht erlaubt!** Sie können mit Ihren Freunden über die Aufgaben sprechen und diskutieren, aber Sie sollten die Aufgaben **individuell** lösen. Plagiate werden nicht toleriert.
- Sie können ChatGPT oder andere Webquellen verwenden, aber Sie dürfen nicht die ganze Frage stellen. Sie sollten in der Lage sein, jeden Schritt Ihres Codes zu erklären.
- Laden Sie alle Dateien, die Sie einreichen möchten, in **eine einzelne .zip-Datei auf Moodle** hoch. Bitte stellen Sie sicher, dass Sie nur .pdf- und .hs-Dateien einreichen.
- Bitte schreiben Sie Ihre **Matrikelnummer**, **Name** und **Gruppennummer** zu jeder Datei, die Sie einreichen.
- Der Name der einzureichenden Datei sollte in der folgenden Reihenfolge angegeben werden:
  Name_Matrikelnummer_Gruppe_GruppenNummer_Übung_ÜbungsNummer.extension.
  Zum Beispiel **AlperenKantarci_1111111_Group_2_Exercise_1.zip**

Aufgabe 1 - Interpreter **(= 10 Punkte)**

In dieser Übung werden Sie einen Interpreter für eine grundlegende Turtle-Grafik-Sprache schreiben. Turtle Graphic war Teil der ursprünglichen Programmiersprache Logo, die 1967 von Wally Feurzeig, Seymour Papert und Cynthia Solomon entwickelt wurde.

Die 'Schildkröte' ('turtle') ist ein Bildschirmcursor (on-screen cursor), der sich auf Befehl über den Bildschirm bewegen und dabei zeichnen kann. Er wird 'Schildkröte' genannt, weil der Bildschirmcursor in einfacher Form als Dreieck oder als Schildkrötensymbol in komplexeren Formen dargestellt werden kann. Unsere Schildkröte (oder unser Stift) kann sich in vier Richtungen bewegen: Norden, Süden, Osten und Westen. Sie hat vier verschiedene Farben: Rot, Blau, Grün und Gelb.

Die folgenden Datentypen werden Ihnen in der Datei main.hs zur Verfügung gestellt. Diese Datei enthält auch die Hauptfunktion für Sie.

```haskell
data Direction = North | South | East | West
    deriving (Show, Eq)

data Color = Red | Blue | Green | Yellow
    deriving (Show, Eq)
```

```haskell
data Command
    = Move Int      -- Move forward by a certain number of steps
    | Turn Direction    -- Turn to face a given direction
    | PenUp             -- Lift the pen (stop drawing)
    | PenDown           -- Lower the pen (start drawing)
    | SetColor Color    -- Set the color of the pen
    deriving (Show, Eq)
```

Für den Interpreter verwenden Sie einen Status ('State'), der alle notwendigen Informationen für den jeweiligen Status enthält.

```haskell
data State = State
    { position :: (Int, Int)
    , direction :: Direction
    , pen :: Bool
    , color :: Color
    }
    deriving (Show, Eq)
```

Sie werden die folgenden Funktionen implementieren, um diese Schildkrötensprache zu interpretieren:

1. Die Funktion 'move' aktualisiert die Position basierend auf der gegebenen Richtung. x ist die Koordinate der horizontalen Achse und der Wert von x erhöht sich, wenn Sie nach Osten gehen. y ist die Koordinate der vertikalen Achse und der Wert von y erhöht sich, wenn Sie nach Süden gehen.

2. Die Funktion 'run' nimmt eine Liste von Befehlen und einen Ausgangszustand ('initial state') an und gibt eine Liste von Liniensegmenten zurück, die die von der Schildkröte hinterlassenen Spuren darstellen.

3. Die Interpret Funktion nimmt eine Liste von Befehlen und einen Ausgangszustand entgegen und gibt das Tupel (Endzustand, Liste der Linien) zurück. Wenn mit dem Befehl keine Linie gezeichnet wird, sollte die Liste leer sein.

Hier sind einige Beispiele für die Ein- und Ausgaben ('inputs & outputs') des Programms:

Beispiel 1

Input

```haskell
cmds = [PenDown, Move 10, Turn East, Move 10, PenUp, SetColor Red, PenDown, Move 10, Move 10]
initialState = State (0, 0) North False Blue
```

Die Spur:

```
[[],[((0,0),(0,-10),Blue)],[],[((0,-10),(10,-10),Blue)],[],[],[],
↪  [((10,-10),(20,-10),Red)],[((20,-10),(30,-10),Red)]]
```

Endgültiger Status:

```haskell
State {position = (30,-10), direction = East, pen = True, color = Red}
```

Beispiel 2

Input

```
cmds = [PenDown, Turn East, Move 50, PenUp, SetColor Blue, PenDown, Move 40, PenUp, SetColor
↪  Red, PenDown, Move 10, Move 20]

initialState = State (20, 20) South False Yellow
```

Die Spur:

```
[[],[],[((20,20),(70,20),Yellow)],[],[],[],[((70,20),(110,20),Blue)],[],[],[],
↪  [((110,20),(120,20),Red)],[((120,20),(140,20),Red)]]
```

Endgültiger Status:

```
State {position = (140,20), direction = East, pen = True, color = Red}
```

1 Aufgabe 2 - Kombinatoren (= 10 Punkte)

1.1 Zeichnen von Formen

```haskell
-- Code to build upon for the exercise
data Point = Point Int Int
data Line = Line Point Point
data Curve = Curve [Point]
data Polygon = Polygon [Point]
data Circle = Circle Point Int  -- Center and radius
data Ellipse = Ellipse Point Point  -- Two foci

data Shape = SPoint Point
           | SLine Line
           | SCurve Curve
           | SPolygon Polygon
           | SCircle Circle
           | SEllipse Ellipse

-- Move a point by dx in the x direction and dy in the y direction
movePoint :: Point -> Int -> Int -> Point
movePoint (Point x y) dx dy = Point (x + dx) (y + dy)

-- Move a shape by dx in the x direction and dy in the y direction
moveShape :: Shape -> Int -> Int -> Shape
moveShape (SPoint p) dx dy = SPoint (movePoint p dx dy)
moveShape (SLine (Line p1 p2)) dx dy = SLine (Line (movePoint p1 dx dy) (movePoint p2 dx dy))
-- ... continue this pattern for Curve, Polygon, Circle, Ellipse
```

1. Vervollständigen Sie die Funktion moveShape, um Kurve, Polygon, Kreis und Ellipse zu ermöglichen.

2. Implementieren Sie die Funktion scaleShape, die eine Form um einen bestimmten Faktor skaliert. Bei Punkt (als geometrische Struktur) hat dieser Vorgang keine Auswirkungen.

3. Implementieren Sie die Funktion rotatePoint, die einen Punkt um den Ursprung um einen bestimmten Winkel (in Grad) dreht. Implementieren Sie anschließend die Funktion rotateShape, mit der eine Form um den Ursprung gedreht wird.