Universität der Bundeswehr
Fakultät für Informatik – Institut 2
Prof. Dr. Wolfgang Hommel
Michael Grabatin, Michael Steinke

WT 2020
Übungsblatt 1

Einführung in die Informatik II

17. und 20.01.2019

1 Basisoperationen

Hinweis: Diese Aufgabe enthält im Wesentlichen nur leicht modifizierte Beispiele aus der Vorlesung. Für diese Aufgaben wird auch **keine Musterlösung** zur Verfügung gestellt.

a) Welche Ausgabe erzeugt der folgende Code-Ausschnitt? Geben Sie die erste und die letzte Ziffer der ausgegebenen Folge an.

```
1 var x = 10
2
3 do {
4     println(x)
5     x -= 1
6 } while(x > 0)
```

b) Welche Ausgabe erzeugt der folgende Code-Ausschnitt? Wenn Sie sich nicht sicher sind, probieren Sie den Code in einem Scala-Worksheet oder einer Scala-REPL selber aus.

```
1  var x = 100
2
3  def incCounter : Int = {
4      x += 1
5      return x
6  }
7
8  def a(u: => Int) = {
9      for(i <- 1 to 10) {
10         println(u)
11     }
12 }
13
14 def b(u: Int) = {
15     for(i <- 1 to 10) {
16         println(u)
17     }
18 }
19
20 a(incCounter)
21 b(incCounter)
22 println(x)
```

c) Vervollständigen Sie die folgenden, mit ____ (mehrere Underscores) markierten, teilweise implementierten Funktionen:

```
1 // Legen Sie ein Array mit 10 Einträgen vom Wert 1 an
```

```
var a = Array.fill(____)(____)

// Diese Funktion soll testen, ob das Array die Länge 0 hat oder nicht
def hasLengthZero(a: _________________): Boolean = {
    if(_________________) {
        return true
    } else {
        return false
    }
}

// Diese Funktion soll alle Elemente des Arrays ausgeben
def printArrayElements(a: Array[Int]): ______ = {
    for(i <- _____________) {
        println(a(i))
    }
}

// Diese Funktion soll ein Array zurückgeben, welches alle Einträge des
// übergebenen Arrays in umgekehrter Reihenfolge enthält.
def reverseArray(a: Array[Int]): Array[Int] = {
    var newArray = ___________________________________
    for(i <- ____________) {

    }
    return _____________
}
```

d) Erstellen Sie eine Funktion, die ein zweidimensionales Array zurück gibt. Diese Funktion soll als Eingabe ein eindimensionales Array an Werten und einem Parameter für die maximale Zeilenlänge des Arrays haben.

```
// Array der Länge 18 mit zufällig gewählten Einträgen
var test : Array[Int] = Array.tabulate___________________
// Ergebnis> test: Array[Int] = Array(51, 76, 0, 28, 59, 33, 77, 54, 10, 33, 33, 8, 1, 32, 6, 96, 82, 63)

def createMatrix(a: Array[Int], maxLength: Int): Array[_____________] = {
    // Anzahl der benötigten Zeilen berechnen
    var rows : Int = Math.ceil(a.length.toDouble / maxLength).toInt

    // Zweidimensionales Array der Größe rows * maxLength anlegen
    // Alle Zellen sollen mit 0 initialisiert werden
    var erg : Array[Array[Int]] = ____________________________

    for(i <- 0 until rows) {
        for(j <- 0 until maxLength) {
            if(i*maxLength + j < a.length) {
                erg___________ = _______
            }
        }
    }
    return erg
```

```
21 }
22
23 createMatrix(test, 5)
24 // Ergebnis> res: Array[Array[Int]] = Array(Array(51, 76, 0, 28, 59),
     Array(33, 77, 54, 10, 33), Array(33, 8, 1, 32, 6), Array(96, 82,
     63, 0, 0))
```

2 Schleifen

Gegeben sei eine ganze Zahl $n >= 5$ und eine wie folgt deklarierte und initialisierte Reihung A:

```
1 var A = Array.range(0, n)
```

Die folgenden vier Programmstücke $(P1)$ - $(P4)$ arbeiten auf A.

Programmstück (P1):

```
1  for (i <- 0 to (k - 1) / 2) {
2    val h = A(i)
3    A(i) = A(k - 1 - i)
4    A(k - 1 - i) = h
5  }
6
7  for (i <- 0 to (n - 1 - k) / 2) {
8    val h = A(k + i)
9    A(k + i) = A(n - 1 - i)
10   A(n - 1 - i) = h
11 }
12
13 for (i <- 0 to (n - 1) / 2) {
14   val h = A(i)
15   A(i) = A(n - 1 - i)
16   A(n - 1 - i) = h
17 }
```

Programmstück (P2):

```
1  for (i <- 0 to (k - 1)) {
2    val h = A(i)
3    A(i) = A(k - 1 - i)
4    A(k - 1 - i) = h
5  }
6
7  for (i <- 0 to (n - 1 - k)) {
8    val h = A(k + i)
9    A(k + i) = A(n - 1 - i)
10   A(n - 1 - i) = h
11 }
12
13 for (i <- 0 to (n - 1)) {
14   val h = A(i)
15   A(i) = A(n - 1 - i)
16   A(n - 1 - i) = h
17 }
```

Programmstück (P3):

```
1    for (i <- 0 to k - 1) {
2      val h = A(0)
3      for (j <- 0 to n - 2) {
4        A(j) = A(j + 1)
5      }
6      A(n - 1) = h
7    }
```

Programmstück (P4):

```
1  for (i <- 0 to n - 1) {
2    val h = A(0)
3    for (j <- 0 to n - 2) {
4      A(j) = A(j + 1)
5    }
6    A(n - 1) = h
7  }
```

a) Führen Sie die Programmstücke für geeignete Testdaten ($n = 5$ und $k = 3$) von Hand aus.

Bei $k = 3$ und $n = 5$ entstehen folgende Ergebnisse:

Programm	Ausgabe
P1	{3, 4, 0, 1, 2}
P2	{0, 1, 2, 3, 4}
P3	{3, 4, 0, 1, 2}
P4	{0, 1, 2, 3, 4}

b) Je zwei der Programmstücke liefern das gleiche Ergebnis. Welche?

- Die Programme $(P1)$ und $(P3)$ verschieben die k ersten Elemente an das Ende von A.
- Die Programme $(P2)$ und $(P4)$ haben (scheinbar) keine Auswirkungen auf A.

c) Erläutern Sie, warum das (auch für andere n und k) so ist, indem Sie erklären, was die Programmstücke und die in ihnen enthaltenen Schleifen bewirken.

Zur Verdeutlichung sei die Reihung wie folgt bildhaft mit seinen Indizes dargestellt:

0	1	..	k-2	k-1	k	k+1	..	n-2	n-1

Programmstück (P1): Jede der drei Schleifen kehrt ein Teil der Reihung um: Die erste Schleife kehrt die ersten k Elemente um, indem sie jeweils das Element mit dem Index i und das Element mit dem Index $k-1-i$ verauscht. Da der umzukehrende Bereich nur zur Hälfte durchlaufen wird, wird sichergestellt, dass jedes Element nur einmal getauscht wird. Nach dem ersten Schleifendurchlauf, sind die Elemente im Vergleich zum Originalzustand wie folgt angeordnet:

k-1	k-2	..	1	0	k	k+1	..	n-2	n-1

Die zweite Schleife kehrt auf analoge Weise die letzten $n-k$ Elemente um:

k-1	k-2	..	1	0	n-1	n-2	..	k+1	k

Die dritte Schleife bewirkt schließlich eine Umkehrung des gesamten Bereiches:

k	k+1	..	n-2	n-1	0	1	..	k-2	k-1

Insgesamt entspricht der Ausführung des Programmstücks das Verschieben (Shift) der ersten k Elemente (Indizes 0 bis $k-1$) an das Ende der Reihung.

Programmstück (P2): Das Programmstück $(P2)$ unterscheidet sich von $(P1)$ darin, dass in den Schleifen nicht jeweils nur der halbe, sondern der ganze Bereich durchlaufen wird. Dadurch wird jedes Element zweimal getauscht und somit in jeder Schleife eine doppelte Umkehrung bewirkt, wodurch der Originalzustand wieder hergestellt wird. Die Reihung bleibt daher von jeder der drei Schleifen unverändert.

Programmstück (P3): Innerhalb der äußeren Schleife werden alle Elemente um einen Index nach links verschoben. Das erste Element (Index 0) wird hinten wieder angehängt. Die äußere Schleife bewirkt die k-malige Durchführung dieses einstelligen Shiftvorganges. Dadurch werden die k ersten Elemente vorne entfernt und hinten angehängt.

Programmstück (P4): Das Programmstück $(P4)$ unterscheidet sich von $(P3)$ darin, dass die äußere Schleife die n-malige Durchführung des einstelligen Shiftvorgangs bewirkt. Dadurch werden n Elemente vorne entfernt und hinten anghängt, was wiederum die ursprüngliche Reihung liefert.