1 Prüfsumme bestimmen

Lösungsidee:

Zuerst abprüfen ob das Array Werte beinhaltet, dannach mithilfe einer while schleife die einzelnen Einträge des Arrays zusammenzählen. Zum Schluss noch die Summe der Einträge durch modulo 256 rechen.

Pseudo-Code Alogrithmus

```
checksumDeterminer (↓integer[] values, ↑integer checksum)
    checksum = 0

    if (values.length > 0)
        integer i = 0
        while (i < values.length)
            checksum += values[i]
            i++
        end while

        checksum = checksum % 256
    end if
end checksumDeterminer
```

Schreibtischtest

Annahme: {10, 11, 30}	checksum	values.length > 0	i	i < values.length
	0	TRUE	0	TRUE
	10		1	TRUE
	21		2	TRUE
	51		3	FALSE
	51			
Annahme: {}	0	FALSE		
	0			
Annahme: {10, -200, 31}	0	TRUE	0	TRUE
	10		1	TRUE
	-190		2	TRUE
	-159		3	FALSE
	97			

2 Tag im Jahr

Lösungsidee:

Zuerst Tag und Jahr Eingabe auf Richtigkeit prüfen (beide sollten größer 0 sein). Beginn der Schaltjahre wird ignoriert, sprich das jahr 4 ist auch ein Schaltjahr. Monats string überprüfung, nur drei chars lang, nicht case sensitive. Wenn alle Eingaben stimmen wird überprüft ob das eingegebene Jahr ein Schaltjahr ist. Wenn es ein Schaltjahr ist wird der Tage eintrag für den Monat Februar von 28 auf 29 geändert. Schlussendlich werden alle Monate bis zum Monatsindex - 1, der bei der Monats Eingabenrüfung ermittelt wird, alle Tage der Monate zusammen gezählt. Dannach wird nur noch die eingegebenen Tage als Schluss Monat addiert.

Pseudo-Code Alogrithmus

```
getDayInYear(↓integer day, ↓string month, ↓integer year, ↑integer dayInYear)
    if (day <= 0 || year <= 0)
        dayInYear = -1
    else
        string[] allMonths = {"jan", "feb", "mär", "apr", "mai", "jun", "jul", "aug", "sep", "okt", "nov", "dez"}

        boolean monthInputOk = false
        integer monthIndex = 0
        integer i = 0
        while (i < allMonths.length)
            if (allMonths[i] == toLower(↓month)) // toLower => convert all chars of the month string to lower chars
                monthInputOk = true
                monthIndex = i
                break // jump out of the while
            end if
            i++
        end while

        if (monthInputOk)
            boolean isLeapYear = false
            leapYearChecker(↓year, ↕isLeapYear)

            integer[] daysInMonths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}
            if (isLeapYear && monthIndex == 1)
                dayInMonths[1] = 29
            end if

            if (day <= daysInMonths(monthIndex))
                dayInYear = 0

                i = 0
                while (i < monthIndex)
                    dayInYear += dayInMonths[i]
                    i++
                end while

                dayInYear += day
            else
                dayInYear = -1
            end if
        else
            dayInYear = -1
        end if
    end if
end getDayInYear
```

Schreibtischtest

string[] allMonths = {"jan", "feb", "mär", "apr", "mai", "jun", "jul", "aug", "sep", "okt", "nov", "dez"}
integer[] daysInMonths = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}

Annahme: 1, "Jan", 2005	dayInYear	day <= 0 || year <= 0	monthInputOk	monthIndex	isLeapYear	isLeapYear && monthIndex == 1	day <= daysInMonths(monthIndex)	i	i < monthIndex
		TRUE	TRUE	0	FALSE	FALSE	TRUE	0	FALSE
	1								
Annahme: 29, "Feb", 2004		TRUE	TRUE	1	TRUE	TRUE	TRUE	0	TRUE
	31							1	FALSE
	60								
Annahme: 29, "Feb", 2005		TRUE	TRUE	1	FALSE	FALSE	FALSE		
	-1								
Annahme: 23, "Juni", 1996		TRUE	FALSE						
	-1								
Annahme: 31, "Sep", 03		TRUE	TRUE	8	FALSE	FALSE	FALSE		
	-1								
Annahme: 31, "Dez", 2020		TRUE	TRUE	11	TRUE	FALSE	TRUE	0	TRUE
	31							1	TRUE
	59							2	TRUE
	90							3	TRUE
	120							4	TRUE
	151							5	TRUE
	181							6	TRUE
	212							7	TRUE
	243							8	TRUE
	273							9	TRUE
	304							10	TRUE
	334							11	FALSE
	365								
Annahme: 0, "Jan", 2020 || 20, "Jan", -10 usw.		FALSE							
	-1								

3 Erweiterter Vergleich von Feldern

Lösungsidee:

Zuerst überprüfen ob die Arrays gleich lang sind. Dannach mit einer Schleife über alle Werte drüber iterieren und bei jeder Iteration überprüfen ob die beiden aktullen Zahlen der Arrays gleich sind (mit berücksichtigung von den Spezialregeln von der Angabe).

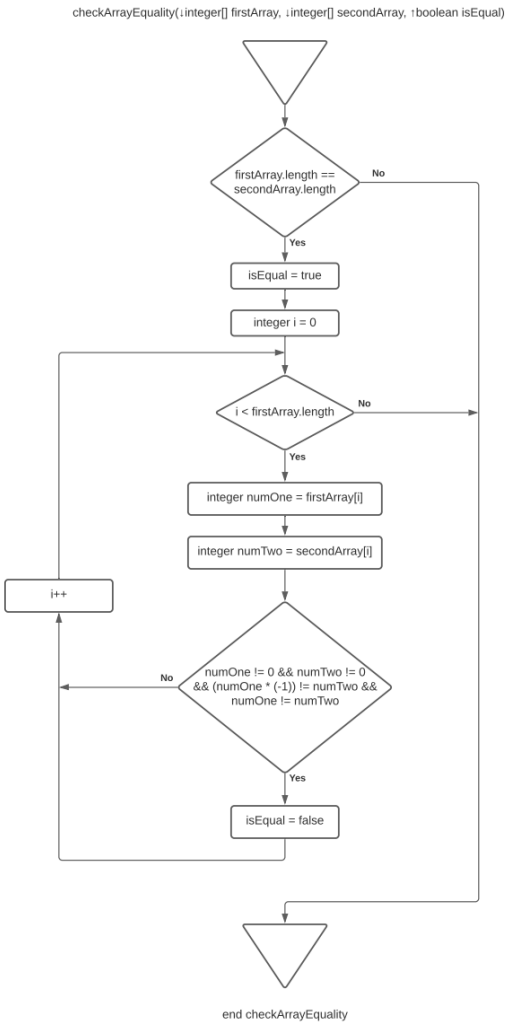
a) Pseudo-Code Alogrithmus

```
checkArrayEquality(↓integer[] firstArray, ↓integer[] secondArray, ↑boolean isEqual)
    if (firstArray.length == secondArray.length)
        isEqual = true

        integer i = 0
        while (i < firstArray.length)
            integer numOne = firstArray[i]
            integer numTwo = secondArray[i]

            if (numOne != 0 && numTwo != 0 && (numOne * (-1)) != numTwo && numOne != numTwo)
                isEqual = false
            end if
            i++
        end while
    else
        isEqual = false
    end if
end checkArrayEquality
```

b) Ablaufdiagramm



Schreibtischtest

Annahme: {10, 0, 2, 7}, {10, -10, -2, 7}	firstArray.length == secondArray.length	isEqual	i	i < firstArray.length	numOne	numTwo	numOne != 0 && numTwo != 0 && (numOne * (-1)) != numTwo && numOne != numTwo
	TRUE	TRUE	0	TRUE	10	10	FALSE
			1	TRUE	0	-10	FALSE
			2	TRUE	2	-2	FALSE
			3	TRUE	7	7	FALSE
			4	FALSE			
Annahme: {10, -5, 8}, {0, 3, 8}	TRUE	TRUE	0	TRUE	10	0	FALSE
			1	TRUE	-5	3	TRUE
		FALSE	2	TRUE	8	8	FALSE
			3	FALSE			
Annahme: {2, 5}, {10} || {}, {11}	FALSE	FALSE					
Annahme: {}, {}	TRUE	TRUE	0	FALSE			

4 Entfernen von negativen Werten

Lösungsidee:

Zuerts überprüfen ob das übergebene Array Werte beinhaltet. Dannach werden die validValues bestimmt und anschließend wird mit einer while über alle Einträge des value Arrays drüber iteriert und dabei abgefragt ob man nicht unnötig iteriert, da schon null values drin stehen können. Wenn in der Iteration der aktuelle Wert des Arrays < 0 ist werden alle Einträge des Arrays um eine Stelle nach vorne geschoben, der letzte Wert des Arrays wird auf null gesetzt und die validValues vom array wird eins weniger. Der Iteratorindex wird aber nur hinauf gezählt wenn bei der Iteration der Wert nicht < 0 ist. Dadurch wird die gleiche stelle des value Arrays nochmal überprüft, da bei negativem Wert alle positionen bei index >= i um eins verschoben werden. Nach der while wird noch zusätzlich überprüft ob der letzte gültige Wert des Arrays postiv ist. Wenn ja werden die validValues um eins erhöht sonst wird der Wert auf null gesetzt.

Pseudo-Code Alogrithmus

```
removeNegativeValues(↕integer[] values, ↑integer amountOfValidValues)
    amountOfValidValues = 0

    if (values.length > 0)
        amountOfValidValues = values.length - 1

        integer i = 0
        while (i < values.length - (values.length - amountOfValidValues))
            if (values[i] < 0)
                integer j = i

                while (j < amountOfValidValues)
                    values[j] = values[j + 1]
                    j++
                end while

                values[amountOfValidValues] = null
                amountOfValidValues--
            else
                i++
            end if
        end while

        if (values[amountOfValidValues] < 0)
            values[amountOfValidValues] = null
        else
            amountOfValidValues++
        end if
    end if
end removeNegativeValues
```

Schreibtischtest

	(values.length - 1)								
Annahme: {1, -4, 3, 5, -1, -2, 8, -5}	amountOfValidValues	integer[] values	values.length > 0	i	i < values.length - (values.length - amountOfValidValues)	values[i] < 0	j	j < amountOfValidValues	values[amountOfValidValues] < 0
	7	{1, -4, 3, 5, -1, -2, 8, -5}	TRUE	0	TRUE	FALSE			
				1	TRUE	TRUE	1	TRUE	
		{1, 3, 3, 5, -1, -2, 8, -5}					2	TRUE	
		{1, 3, 5, 5, -1, -2, 8, -5}					3	TRUE	
		{1, 3, 5, -1, -1, -2, 8, -5}					4	TRUE	
		{1, 3, 5, -1, -2, -2, 8, -5}					5	TRUE	
		{1, 3, 5, -1, -2, 8, 8, -5}					6	TRUE	
		{1, 3, 5, -1, -2, 8, -5, -5}					7	FALSE	
	6	{1, 3, 5, -1, -2, 8, -5, null}			TRUE	FALSE			
				2	TRUE	FALSE			
				3	TRUE	TRUE	3	TRUE	
		{1, 3, 5, -2, -2, 8, -5, null}					4	TRUE	
		{1, 3, 5, -2, 8, 8, -5, null}					5	TRUE	
		{1, 3, 5, -2, 8, -5, -5, null}					6	FALSE	
	5	{1, 3, 5, -2, 8, -5, null, null}			TRUE	TRUE	3	TRUE	
		{1, 3, 5, 8, 8, -5, null, null}					4	TRUE	
		{1, 3, 5, 8, -5, -5, null, null}					5	FALSE	
	4	{1, 3, 5, 8, -5, null, null, null}			TRUE	FALSE			
				4	FALSE				TRUE
		{1, 3, 5, 8, null, null, null, null}							
Annahme: {4, 0, 10}	2	{4, 0, 10}	TRUE	0	TRUE	FALSE			
				1	TRUE	FALSE			
				2	FALSE				FALSE
	3	{4, 0, 10}							
Annahme: {}	0	{}	FALSE						
Annahme: {-3, -10, -2}	2	{-3, -10, -2}	TRUE	0	TRUE	TRUE	0	TRUE	
		{-10, -10, -2}					1	TRUE	
		{-10, -2, -2}					2	FALSE	
	1	{-10, -2, null}			TRUE	TRUE	0	TRUE	
		{-2, -2, null}					1	FALSE	
	0	{-2, null, null}			FALSE				TRUE
		{null, null, null}							