Aufgabe 1

```
Gedanke: Summe bilden und dann mod256 rechen
```

```
1  checksum(Integer arraylength, Integer array[]): Integer
2      Integer result = 0
3      for (Integer i=0;i<arraylength;i+=1)
4          result += array[i]
5      end
6      return result%256
7  end
```

Test: Array[1,2,3,4,5] arraylength = 5

Schritt	Zeile	Bedingung t|f	i	result
1	2			0
2	3	t	0	
3	4			1
4	5		1	
5	3	t		
6	4			3
7	5		2	
8	3	t		
9	4			6
10	5		3	
11	3	t		
12	4			10
13	5		4	
14	3	t		
15	4			15
16	5		5	
17	3	f		
18	6			15%256

Aufgabe 2

Gedanke: gregorianischer Kalender: Feb hat im Schaltjahr 29 Tage, Anzahl der Tage des Monats und Monatsname im Feld speichern dann kann man sie über Indices erreichen kann und nicht viele if's benötigt. Wenn es ein Schaltjahr ist (alter Code) dann Anzahl der Tage auf 29 setzten. Überprüfen, ob Monat existiert, indem man die Einträge im Feld mit dem String Monat den man bekommt vergleicht. Wenn der Monat angenommen wird dann schauen ob der Tag den man bekommt kleiner als das Maximum des Monats ist und größer Null, dann kann man den Wert des Tages auf die tagImJahr Variable addieren. Wenn das nicht zutrifft dann halt Fehler ausgeben, genauso wenn es den Monat nicht gibt. Und wenns der Monat noch nicht war und noch nicht Dezember war dann halt die Tage des Monats aufaddieren.

Für das zweite Codebeispiel kann der Wechsel vom julianischen Kalender auf den gregorianischen Kalender berücksichtig werden. Dazu ist zu wissen, dass der 5-15.Oktober.1582 ausgelassen wurden. Zusätzlich gab es zur Einführung vom julianischen Kalender einen Fehler in der Schaltjahreszählung

und verstanden statt der alle 4 Jahre ein Schaltjahr, dass sie nach der Inklusivrechnung somit alle 3 Jahre den 29.Februar eingeführt haben. Kaiser Augustus hat diesen Fehler beglichen, da er die Schaltjahre 5 v.Chr, 1 v.Chr ,4 n.Chr,8 n.Chr. ausfallen lassen hat. Augustus hat dann die Schaltjahresreform von Cäsar umgesetzt und ab dem Jahr 0 gilt jahr%4==0 ist ein Schaltjahr.

Spezifikation

Jahreszahlen mit vor Chr. Werden mit <-Jahr> geschrieben

Jahreszahlen können 0 bis 5 Stellen haben.

Im zweiten Codebeispiel sind die Jahreszahlen von]-55, 65.536 [definiert.

> @param: boolean typejulianbefore1582 true, ermöglicht die Verwendung vom julianischen Kalender

Code 1: nur gregorianisch:

```
1  public Integer validdatetest(Integer day, String month, Integer year)
2      boolean valid = false
3      Integer[] daysInMonth = {0,31,28,31,30,31,30,31,31,30,31,30,31}
4      Integer i = 1
5      String[] namesOfMonths =
6  {"","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"
7  }
8      Integer dayInYear = 0
9      if (year % 4==0)
10         daysInMonth[2]+=1
11         if (year % 100==0)
12             daysInMonth[2]-=1
13             if (year % 400==0)
14                 daysInMonth[2]+=1
15             end
16         end
17     end
18     i = 1
19     while (i < 13)
20         if (namesOfMonths[i] == month)
21             if ((daysInMonth[i] >= day)&&(day > 0))
22                 dayInYear += day
23                 valid = true
24             end
25         end
26         if (valid != true)
27             dayInYear = dayInYear + daysInMonth[i]
28         end
29         i+=1
30     end
31     if (valid == false){
32             printf("ERROR - THE INSERT DATE IS NOT VALID")
33             dayInYear = -1
34         end
35     return dayInYear
36 end
```

Code 2:

```
validdatetest(Integer day, String month, Integer year, boolean typejulianbefore1582):Integer
    boolean isLeapYear = false
    boolean valid = false
    Integer[] daysInMonth = {0,31,28,31,30,31,30,31,31,30,31,30,31}
    Integer i = 1
    String[] namesOfMonths =
{"","Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"}
    Integer dayInYear = 0
    if (year % 4==0)
        daysInMonth[2]+=1
        if (year % 100==0)
            daysInMonth[2]-=1
            if (year % 400==0)
                daysInMonth[2]+=1
            end
        end
    end
    if ((year > 1582)|||(typejulianbefore1582 == false))
        i = 1
        while (i < 13)
            if (namesOfMonths[i] == month)
                if ((daysInMonth[i] >= day)&&(day > 0))
                    dayInYear += day
                    valid = true
                end
            end
            if (valid != true)
                dayInYear = dayInYear + daysInMonth[i]
            end
            i+=1
        end
    else
        if (year == 1582)
            i = 1
           daysInMonth[10]-=11
            while (i < 13)
                if (namesOfMonths[i] == month)
                    if ((daysInMonth[i] >= day)&&(day > 0)&&(i!=10))
                        dayInYear += day
                        valid = true
                    else
                        if ((day > 0)&&(daysInMonth[10]<=day)&&(day < 5)&& (day > 14))
                            dayInYear += day
                            valid = true
                        else
                            if (namesOfMonths[10]==month)
                                printf("The 5. to 15. of October 1582 were left out to
compensate for mistakes in calculations of the Julian calendar ")
                            end

                        end
                    end
                end
                if(valid!=true){
                    dayInYear += daysInMonth[i]
                end
                i+=1
            end
        else
            if ((year < 1582)&& (year>0))
                if ((year%4==0) && (year!=4) && (year!=8))
                    daysInMonth[2]= 29
                end
                i = 1
                while (i < 13){
                    if (namesOfMonths[i] == month){
                        if ((daysInMonth[i] >= day)&&(day > 0))
                            dayInYear += day
                            valid = true
                        end
                    end
                    if(valid!=true){
                        dayInYear += daysInMonth[i]
                    end
                    i+=1
                end
            else
                if (year > -45){
```

```
79                      if ((year % 3==0)&&(year!=1)&&(year!=5))
80                          daysInMonth[2]= 29
81                      end
82                      i = 1
83                      while (i < 13){
84                          if (namesOfMonths[i] == month){
85                              if ((daysInMonth[i] >= day)&&(day > 0))
86                                  dayInYear += day
87                              end
88                          end
89                          if(valid!=true){
90                              dayInYear += daysInMonth[i]
91                          end
92                          i+=1
93                      end
94                  else
95                      printf("This year is not defined")
96                  end
97              end
98          end
99      end
100     if (valid == false){
101             printf("ERROR - THE INSERT DATE IS NOT VALID")
102             dayInYear = -1
103         end
104     return dayInYear
105 end
```

Test durchgeführt mit Java-Umsetzung, zusätzliche Datei mit Java-Code
Console für Code 2:

NEW DATE:29Feb12julKtrueSTART

DATE:29Feb12 DONE - return value:60

29.Feb.12 ist der 60Tag im Jahr, BSP für Schaltjahr %4

NEW DATE:29Feb12julKfalseSTART

DATE:29Feb12 DONE - return value:60

29.Feb.12 ist der 60Tag im Jahr, BSP für Schaltjahr %4

NEW DATE:29Feb100julKtrueSTART

DATE:29Feb100 DONE - return value:60

29.Feb.100 ist der 60Tag im Jahr, BSP für Schaltjahr %100

NEW DATE:29Feb100julKfalseSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb100 DONE - return value:-1

29.Feb.100 ist der -1Tag im Jahr, BSP für Schaltjahr %100

NEW DATE:29Feb2000julKtrueSTART

DATE:29Feb2000 DONE - return value:60

29.Feb.2000 ist der 60Tag im Jahr, BSP für Schaltjahr %400

NEW DATE:29Feb2000julKfalseSTART

DATE:29Feb2000 DONE - return value:60

29.Feb.2000 ist der 60Tag im Jahr, BSP für Schaltjahr %400

NEW DATE:29Feb2001julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb2001 DONE - return value:-1

29.Feb.2001 ist der -1Tag im Jahr, BSP für falsche Schaltjahr

NEW DATE:29Feb2001julKfalseSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb2001 DONE - return value:-1

29.Feb.2001 ist der -1Tag im Jahr, BSP für falsche Schaltjahr

NEW DATE:29Feb1423julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb1423 DONE - return value:-1

29.Feb.1423 ist der -1Tag im Jahr, BSP für falsche Schaltjahr

Testset: Ausgelasse Tage in julianischer Kalender

NEW DATE:29Feb-11julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb-11 DONE - return value:-1

29.Feb.-11 ist der -1Tag im Jahr, BSP für Schaltjahr falsche Berechnung

NEW DATE:29Feb-5julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb-5 DONE - return value:-1

29.Feb.-5 ist der -1Tag im Jahr, BSP für Schaltjahr: ausgelassen von Augustus

NEW DATE:29Feb-1julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb-1 DONE - return value:-1

29.Feb.-1 ist der -1Tag im Jahr, BSP für Schaltjahr: ausgelassen von Augustus

NEW DATE:29Feb4julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb4 DONE - return value:-1

29.Feb.4 ist der -1Tag im Jahr, BSP für Schaltjahr: ausgelassen von Augustus

NEW DATE:29Feb8julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:29Feb8 DONE - return value:-1

29.Feb.8 ist der -1Tag im Jahr, BSP für Schaltjahr: ausgelassen von Augustus

NEW DATE:13Oct1582julKtrueSTART

The 5. to 15. of October 1582 were left out to compensate for mistakes in calculations of the Julian calendar

ERROR - THE DATE IS NOT VALID

DATE:13Oct1582 DONE - return value:-1

13.Oct.1582 ist der -1Tag im Jahr, BSP für ausgelassene Tage im Jahr 1582

TESTSET: FALSCHE Eingaben

NEW DATE:31Jun2017julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:31Jun2017 DONE - return value:-1

31.Jun.2017 ist der -1Tag im Jahr, BSP für falsche Eingabe: Tag

NEW DATE:31Jun2017julKfalseSTART

ERROR - THE DATE IS NOT VALID

DATE:31Jun2017 DONE - return value:-1

31.Jun.2017 ist der -1Tag im Jahr, BSP für falsche Eingabe: Tag

NEW DATE:31Jun1582julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:31Jun1582 DONE - return value:-1

31.Jun.1582 ist der -1Tag im Jahr, BSP für falsche Eingabe: Tag

NEW DATE:31Jun-23julKfalseSTART

ERROR - THE DATE IS NOT VALID

DATE:31Jun-23 DONE - return value:-1

31.Jun.-23 ist der -1Tag im Jahr, BSP für falsche Eingabe: Tag

NEW DATE:13ANA2017julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:13ANA2017 DONE - return value:-1

11.ANA.2017 ist der -1Tag im Jahr, BSP für falsche Eingabe: Monat

NEW DATE:13ANA1582julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:13ANA1582 DONE - return value:-1

11.ANA.1582 ist der -1Tag im Jahr, BSP für falsche Eingabe: Monat

NEW DATE:31Jun2017julKtrueSTART

ERROR - THE DATE IS NOT VALID

DATE:31Jun2017 DONE - return value:-1

11.ANA.-23 ist der -1Tag im Jahr, BSP für falsche Eingabe: Monat

Aufgabe 3

a) Pseudocode

```
Idee: Die Felder an ihren Indices vergleichen nach den Vorschriften
3=-3 und -3=3 kann durch(-1*a==b)OR(-1*b==a) gelöst werden. Wenn a=b gilt
dann wird
```

```
1  valuesComperison(Integer a1[],Integer a2[]):boolean
2      Boolean result = false;
3      if (a1[].length == a2[].length)
4          for (Integer i = 0; i < a1.length; i+=1)
5              if ((a1[i]== a2[i])OR(a1[i]==(a2[i]*-1)) OR(a2[i]==(a1[i]*-1))
6              OR(a1[i]== 0)OR(a2[i]== 0))
7                  result = true
8              else
9                  result = false
10                 i = a1[].length
11             end -if
12         end -for
13     end -if
14     return result
15 end -valuesComperison
```

Test a1[0,2,-3] a2[1,-2,3] a1.length = 3 = a2[].length

Schritt	Zeile	Bedingung t|f	i	result
1	2			false
2	3	t		
3	4	t	0	
4	5,6	t		
5	7			true
6	12		1	
7	4	t		
8	5,6	t		
9	7			true
10	12		2	
11	4	t		
12	5,6	t		
13	7			true
14	12		3	
15	7	f		
16	14			true

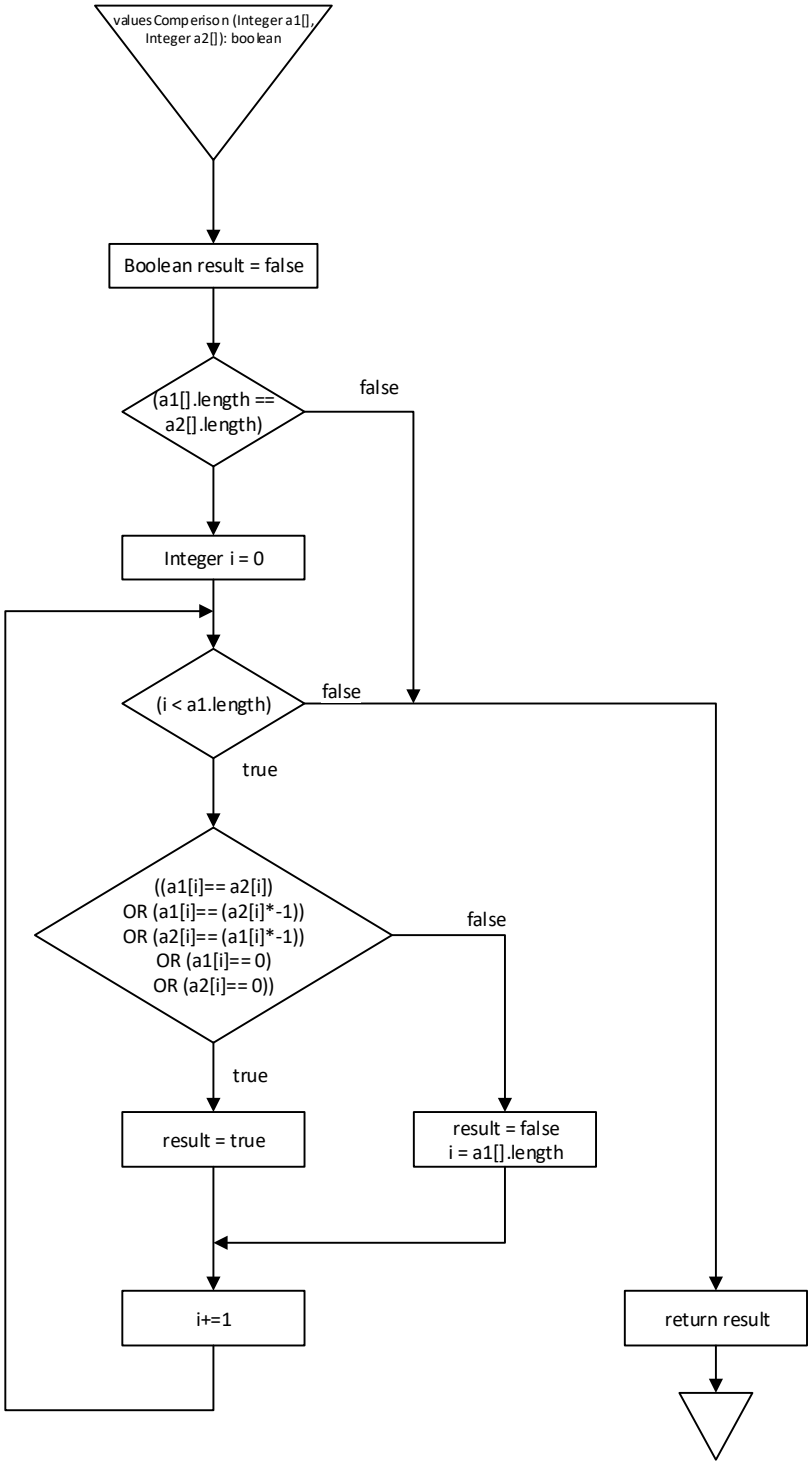
Test a1[].lenght ≠ a2[].lenght

Schritt	Zeile	Bedingung t|f	i	result
1	2			false
2	3	f		
3	14			false

Test a1[2,1,2], a2[2,2,2] a1[].lenght =3= a2[].lenght

Schritt	Zeile	Bedingung t|f	i	result
1	2			false
2	3	t		
3	4	t	0	
4	5,6	t		
5	7			true
6	12		1	
7	4	t		
8	5,6	f		
9	9			false
10	10			
11	12		3	
12	4	f		
13	14			false

b)Ablaufdiagramm



Aufgabe 4

1

```
1   deleteNegativeValues(Integer a1[]):Integer[],Integer
2       Integer a2[a1[].length]={0}
3       Integer nextEmpty=0
4       validDigits = 0
5       Integer i = 0
6       while ((i < a1[].length)AND(a1[].length != 0))
```

```
7           if (a1[i]>=0)
8                   validDigits +=1
9                   a2[nextEmpty] = a1[i]
10                  nextEmpty+=1
11          end -if
12          i+=1
13      end -while
14      return a2[],validDigits
15  end -deleteNegativeValues
```

						a1	a1	a1	a2	a2	a2
Schritt	Zeile	Bedingung	i	validDigits	nextEmpty	[0]	[1]	[2]	[0]	[1]	[2]
						1	-1	3			
	2								0	0	0
	3				0						
	4			0							
	5		0								
	6	t									
	7	t									
	8			1							
	9								1		
	10										
	12		1								
	6	t									
	7	f									
	12		2								
	6	t									
	7	t									
	8			2							
	9									3	
	12		3								
	6	f									
	14	return									