Aufgabe 1

Als Eingang dient ein Integer Array mit unbestimmter Größe. Der Ausgabewert ist ein Integer. Die einzelnen Elemente eines Arrays werden summiert und die Summe anschließend durch Modulo 256 geteilt. Der Rest dieser Division ist die Checksum.

```
1.  checksum(↓integer numbers[], ↑integer cksm)
2.    integer sum = 0
3.
4.    for (i=numbers.size() -1; i>=0; i--)do
5.          sum += numbers[i]
6.    end
7.    cksm = sum % 256
8.  end
```

numbers = { 0,4,6,400,-10 }

sum	cksm	number[i]	i	Abfrage	Befehl
0	0	-10	4	i >= 0	-
0	0	-10	4	true	-
0	0	-10	4	-	sum += numbers[i]
-10	0	400	3	i >= 0	-
-10	0	400	3	true	-
-10	0	400	3	-	sum += numbers[i]
390	0	6	2	i >= 0	-
390	0	6	2	true	-
390	0	6	2	-	sum += numbers[i]
396	0	4	1	i >= 0	-
396	0	4	1	true	-
396	0	4	1	-	sum += numbers[i]
400	0	0	0	i >= 0	-
400	0	0	0	true	-
400	0	0	0	-	sum += numbers[i]
400	144	-	-	-	sum % 256

Aufgabe 2

```
howmuchdays(↓string date,↑integer result)
    string splitteddate[3] = {"","",""}
    integer date_length = date.length()
    integer i = 0
    integer n = 0

    while (i < date_length) do
        if (date[i] == ' ') then
            n := n + 1
        else
            splitteddate[n] = splitteddate[n] + date[i]
        end
        i = i + 1
    end

    if (numbercheck(splitteddate[1]) == true && numbercheck(splitteddate[3]) == true ) then
        result := daycalc(atoi(splitteddate[1]),splitteddate[2],atoi(splitteddate[3] ))
    else
        result := -1
    end
end

daycalc(↓unsigned integer days,↓string month,↓unsigned integer year,↑integer days)
    const string month_str[12] = {"JAN","FEB","MAE","APR","MAI","JUN","JUL","AUG","S EP","OKT","NOV","DEZ"}
    integer month_day[12] = {31,28,31,30,31,30,31,31,30,31,30,31}
    integer numberofmonth := -1;
    integer sumofdays := 0;

    for (int i = 0; i < 12 && numberofmonth == -1; i++) do
        if (month == month_str[i]) then
            numberofmonth = i
            month_day[1] += leapyeardetermination(year)
        end
    end
    if (numberofmonth != -1 && days > 0 && days <= month_day[numberofmonth]) then
        for (integer i = 0; i < numberofmonth;i++) do
            sumofdays += month_day[i]
        end
        sumofdays += days
    else
        sumofdays := -1
    end
    days := sumofdays
end

leapyeardetermination(↓integer Year, ↑integer Result)
    Result := 1
    if (Year < 100) then
        Year = Year + 2000
    end
    if ((Year % 4) == 0) then
        if ((Jahr % 100) == 0)
            if ((Jahr % 100) == 0)
                Result := 0
            end
        end
    else
        Result := 0
    end
end
```

```
61.
62. numbercheck(↓string number, ↑boolean result)
63.     const char numbers[10] = { '0','1','2','3','4','5','6','7','8','9'};
64.     boolean numberchecked = false
65.     result := true
66.     for (int i = number.length() -1; i >= 0; i--) do
67.         numberchecked := false
68.         for (int n = 0; n < 10; n++) do
69.             if (number[i] == numbers[n]) then
70.                 numberchecked := true
71.             end
72.         end
73.         if (numberchecked == false) then
74.             result := false
75.         end
76.     end
77. end
```

Schreibtischtest „numbercheck“

number = “2004”

result	numberchecked	i	Ueberprüfung	Befehl
1	0	-	-	-
1	0	-	-	i = number.length() -1
1	0	3	i >= 0	-
1	0	3	true	numberchecked := false
1	0	3	number[i] == numbers[n]	(For Schleife)
1	0	3	true bei n=4	numberchecked := true
1	1	3	numberchecked == false	-
1	1	3	false	-
1	1	3	i >= 0	-
1	1	2	true	numberchecked := false
1	0	2	number[i] == numbers[n]	(For Schleife)
1	0	2	true bei n=0	numberchecked := true
1	1	2	numberchecked == false	-
1	1	2	false	-
1	1	2	i >= 0	-
1	1	1	true	numberchecked := false
1	0	1	number[i] == numbers[n]	(For Schleife)
1	0	1	true bei n=0	numberchecked := true
1	1	1	numberchecked == false	-
1	1	1	false	-
1	1	1	i >= 0	-
1	1	0	true	numberchecked := false
1	0	0	number[i] == numbers[n]	(For Schleife)
1	0	0	true bei n=2	numberchecked := true
1	1	0	numberchecked == false	-
1	1	0	false	-
1	1	0	i >= 0	-
1	1	-	false	END

number = „R245“

result	numberchecked	i	Ueberprüfung	Befehl
1	0	-	-	-
1	0	-	-	i = number.length() -1
1	0	3	i >= 0	-
1	0	3	true	numberchecked := false
1	0	3	number[i] == numbers[n]	(For Schleife)
1	0	3	true bei n=5	numberchecked := true
1	1	3	numberchecked == false	-
1	1	3	false	-
1	1	3	i >= 0	-
1	1	2	true	numberchecked := false
1	0	2	number[i] == numbers[n]	(For Schleife)
1	0	2	true bei n=4	numberchecked := true
1	1	2	numberchecked == false	-
1	1	2	false	-
1	1	2	i >= 0	-
1	1	1	true	numberchecked := false
1	0	1	number[i] == numbers[n]	(For Schleife)
1	0	1	true bei n=2	numberchecked := true
1	1	1	numberchecked == false	-
1	1	1	false	-
1	1	1	i >= 0	-
1	1	0	true	numberchecked := false
1	0	0	number[i] == numbers[n]	(For Schleife)
1	0	0	numberchecked == false	-
1	0	0	true	result := false
0	0	0	i >= 0	-
0	0	-	false	END

number = „“

result	numberchecked	i	Ueberprüfung	Befehl
1	0	i	-	-
1	0	i	-	i = number.length() -1
1	0	i	i >= 0	-
1	0	-	false	END

howmuchdays("10 JAN 2006") splitteddate == splitd

splitd[0]	splitd[1]	splitd[2]	i	n	Überprüfung	Befehl
			0	0	i < date_length	-
			0	0	true	-
			0	0	date[i] == ' '	-
			0	0	false	splitd[n] += date[i]
1			1	0	-	i := i+1
1			1	0	i < date_length	-
1			1	0	true	-
1			1	0	date[i] == ' '	-
1			1	0	false	splitd[n] += date[i]
10			2	0	-	i := i+1
10			2	0	i < date_length	-
10			2	0	true	-
10			2	0	date[i] == ' '	-
10			2	0	true	n := n + 1
10			3	1	-	i := i+1
10			3	1	i < date_length	-
10			3	1	true	-
10			3	1	date[i] == ' '	-
10			3	1	false	splitd[n] += date[i]
10	J		4	1	-	i := i+1
10	J		4	1	i < date_length	-
10	J		4	1	true	-
10	J		4	1	date[i] == ' '	-
10	J		4	1	false	splitd[n] += date[i]
10	JA		5	1	-	i := i+1
10	JA		5	1	i < date_length	-
10	JA		5	1	true	-
10	JA		5	1	date[i] == ' '	-
10	JA		5	1	false	splitd[n] += date[i]
10	JAN		6	1	-	i := i+1
10	JAN		6	1	i < date_length	-
10	JAN		6	1	true	-
10	JAN		6	1	date[i] == ' '	-
10	JAN		6	1	true	n := n + 1
10	JAN		7	2	-	i := i+1
10	JAN		7	2	i < date_length	-
10	JAN		7	2	true	-
10	JAN		7	2	date[i] == ' '	-
10	JAN		7	2	false	splitd[n] += date[i]
10	JAN	2	8	2	-	i := i+1
10	JAN	2	8	2	i < date_length	-
10	JAN	2	8	2	true	-
10	JAN	2	8	2	date[i] == ' '	-
10	JAN	2	8	2	false	splitd[n] += date[i]
10	JAN	20	9	2	-	i := i+1
10	JAN	20	9	2	i < date_length	-
10	JAN	20	9	2	true	-
10	JAN	20	9	2	date[i] == ' '	-
10	JAN	20	9	2	false	splitd[n] += date[i]
10	JAN	200	10	2	-	i := i+1
10	JAN	200	10	2	i < date_length	-
10	JAN	200	10	2	true	-
10	JAN	200	10	2	date[i] == ' '	-
10	JAN	200	10	2	false	splitd[n] += date[i]
10	JAN	2006	11	2	-	i := i+1
10	JAN	2006	11	2	i < date_length	-
10	JAN	2006	11	2	false	-
10	JAN	2006	11	2	numbercheck for [0] && [2]	-
10	JAN	2006	11	2	false	result := daycalc(. . .)

howmuchdays("10 FF2006") splitteddate == splitd

splitd[0]	splitd[1]	splitd[2]	i	n	Überprüfung	Befehl
			0	0	i < date_length	-
			0	0	true	-
			0	0	date[i] == ' '	-
			0	0	false	splitd[n] += date[i]
1			1	0	-	i := i+1
1			1	0	i < date_length	-
1			1	0	true	-
1			1	0	date[i] == ' '	-
1			1	0	false	splitd[n] += date[i]
10			2	0	-	i := i+1
10			2	0	i < date_length	-
10			2	0	true	-
10			2	0	date[i] == ' '	-
10			2	0	true	n := n + 1
10			3	1	-	i := i+1
10			3	1	i < date_length	-
10			3	1	true	-
10			3	1	date[i] == ' '	-
10			3	1	false	splitd[n] += date[i]
10	F		4	1	-	i := i+1
10	F		4	1	i < date_length	-
10	F		4	1	true	-
10	F		4	1	date[i] == ' '	-
10	F		4	1	false	splitd[n] += date[i]
10	FF		5	1	-	i := i+1
10	FF		5	1	i < date_length	-
10	FF		5	1	true	-
10	FF		5	1	date[i] == ' '	-
10	FF		5	1	false	splitd[n] += date[i]
10	FF2		6	1	-	i := i+1
10	FF2		6	1	i < date_length	-
10	FF2		6	1	true	-
10	FF2		6	1	date[i] == ' '	-
10	FF2		6	1	false	splitd[n] += date[i]
10	FF20		7	1	-	i := i+1
10	FF20		7	1	i < date_length	-
10	FF20		7	1	true	-
10	FF20		7	1	date[i] == ' '	-
10	FF20		7	1	false	splitd[n] += date[i]
10	FF200		8	1	-	i := i+1
10	FF200		8	1	i < date_length	-
10	FF200		8	1	true	-
10	FF200		8	1	date[i] == ' '	-
10	FF200		8	1	false	splitd[n] += date[i]
10	FF2006		9	1	-	i := i+1
10	FF2006		9	1	i < date_length	-
10	FF2006		9	1	false	-
10	FF2006		9	1	numbercheck for [0] && [2]	-
10	FF2006		9	1	false	result := daycalc(. . .)

daycalc(20, "APR", 2006)

numberofmonth	sumofdays	month_day[1]	i	Überprüfung	Befehl
-1	0	28	-	i < 12 && numberofmonth == -1	i:=0
-1	0	28	0	true	-
-1	0	28	0	month == month_str[i]	-
-1	0	28	-	-	i:= i + 1
-1	0	28	-	i < 12 && numberofmonth == -1	-
-1	0	28	1	true	-
-1	0	28	1	month == month_str[i]	-
-1	0	28	-	-	i:= i + 1
-1	0	28	-	i < 12 && numberofmonth == -1	-
-1	0	28	2	true	-
-1	0	28	2	month == month_str[i]	-
-1	0	28	-	-	i:= i + 1
-1	0	28	-	i < 12 && numberofmonth == -1	-
-1	0	28	3	true	-
-1	0	28	3	month == month_str[i]	-
-1	0	28	3	true	numberofmonth := i
3	0	28	-	-	i:= i + 1
3	0	28	-	i < 12 && numberofmonth == -1	-
3	0	28	-	false	-
3	0	28	-	numberofmonth != -1 && days > 0 && days <= month_day[numberofmonth]	-
3	0	28	-	true	
3	0	28	-	i < 12 && numberofmonth == -1	FOR (i:=0; i<numberofmonth; i++)
3	31	28	0	-	sumofdays += month_day[i]
3	59	28	1	-	sumofdays += month_day[i]
3	90	28	2	-	sumofdays += month_day[i]
3	90	28	-	-	sumofdays += days
3	110	28	-	days := sumofdays	END

daycalc(0, "FEB", 2004)

numberofmonth	sumofdays	month_day[1]	i	Überprüfung	Befehl
-1	0	28	-	i < 12 && numberofmonth == -1	i:=0
-1	0	28	0	true	-
-1	0	28	0	month == month_str[i]	-
-1	0	28	-	-	i:= i + 1
-1	0	28	-	i < 12 && numberofmonth == -1	-
-1	0	28	1	true	-
-1	0	28	1	month == month_str[i]	-
-1	0	28	1	true	numberofmonth := i
1	0	28	1	-	month_day[1] += leapyeardetermination(year)
1	0	29	-	-	i:= i + 1
1	0	29	-	i < 12 && numberofmonth == -1	-
1	0	29	-	false	-
1	0	29	-	numberofmonth != -1 && days > 0 && days <= month_day[numberofmonth]	-
1	0	29	-	false	sumofdays := -1
1	-1	29	-	days := sumofdays	END

```
daycalc(30, "MAE", 2004)
```

number ofmonth	sum ofdays	month _day[1]	i	Überprüfung	Befehl
-1	0	28	-	i < 12 && numberofmonth == -1	i:=0
-1	0	28	0	true	-
-1	0	28	0	month == month_str[i]	-
-1	0	28	-	-	i:= i + 1
-1	0	28	-	i < 12 && numberofmonth == -1	-
-1	0	28	1	true	-
-1	0	28	1	month == month_str[i]	-
-1	0	28	-	-	i:= i + 1
-1	0	28	-	i < 12 && numberofmonth == -1	-
-1	0	28	2	true	-
-1	0	28	2	month == month_str[i]	-
-1	0	28	2	true	numberofmonth := i
2	0	28	2	-	month_day[1] += leapyeardetermination(year)
2	0	29	-	-	i:= i + 1
2	0	29	-	i < 12 && numberofmonth == -1	-
2	0	29	-	false	-
2	0	29	-	numberofmonth != -1 && days > 0 && days <= month_day[numberofmonth]	-
2	0	29	-	true	
2	0	29	-	i < 12 && numberofmonth == -1	FOR (i:=0; i<numberofmonth; i++)
2	31	29	0		sumofdays += month_day[i]
2	60	29	1		sumofdays += month_day[i]
2	60	29	-		sumofdays += days
2	90	29	-	days := sumofdays	END

Der Nutzer gibt einen String im folgendem Format ein „dd mmm yyyy“. Jahre die mit zwei Ziffern abgekürzt wurden (z.B. „06“) werden automatisch ins 21. Jahrhundert gesetzt (z.B. „2006“). Der Monat besteht aus drei großgeschriebenen Buchstaben (JAN,FEB,MAE,APR,MAI,JUN,JUL,AUG,SEP,OKT,NOV,DEZ) . Der Tag enthält eine oder zwei Ziffern. Tag, Monat und Jahr sind jew. durch ein Leerzeichen zu trennen.

Im ersten Schritt wird die Funktion *„howmuchdays“* gestartet die den oben genannten String erhält. Diese splittet den String anhand der Leerzeichen in seine einzelnen Bestandteile. Anschließend wird überprüft ob die gesplitteten Strings für Tag und Jahr ausschließlich Nummern enthalten. Falls nicht wird ein -1 zurückgegeben. Falls sie nur Zahlen enthalten werden die Strings in ein Integer gecastet und die Funktion *„daycalc*“ aufgerufen die alle Bestandteile des Datums bekommt.

Die Funktion daycal beinhaltet zwei Arrays mit den Monatskürzeln und den jeweiligen Tagen dazu. Es wird überprüft ob der übergebene Monat Bestandteil des Monatskürzel Arrays ist. Falls Ja wird der Index in einer Variable gespeichert und überprüft ob das Jahr ein Schaltjahr ist. Falls es ein Schaltjahr ist wird dem Monat_Tag Array bei Februar ein Tag hinzugezählt. Zum Schluss wird überprüft ob der übergebene Tag in den Monat passt (tag > 0 und tag <= tage im Monat). Wenn dies nicht zutrifft gibt die Funktion ein -1 zurück. Falls alles passt werden nun anhand der oben hinzugefügten Index Variablen die Monate bestimmt, die vergangen sind. Diese werden nun alle zusammengerechnet und anschließend noch die übergebenen Tage hinzuaddiert. Die summierten Tage werden nun zurückgegeben. Die Funktion *„howmuchdays“* gibt diese nun ebenfalls zurück.

Aufgabe 3

Um zwei Arrays auf Gleichheit zu überprüfen wird als allererstes die Größe der beiden Arrays bestimmt und miteinander verglichen. Falls diese identisch sind wird jedes Feld einzeln miteinander verglichen. Um Minuszahlen zu eliminieren werden die Zahlen ^2 genommen. Falls sie nicht gleich sind wird noch überprüft ob sie 0 sind. Eine 0 führt zu einer Ausnahme.

Zurückgegeben wird ein boolescher Wert.

a)

```
1.  arrayequally(↓integer numbers0[],↓integer numbers1[], ↑bool equally)
2.    equally := true
3.
4.    if (numbers0.size() != numbers1.size()) then
5.      equally := false
6.    else
7.      for(integer i = numbers0.size(); i >= 0 && equally == true; i-- ) do
8.          if ((numbers0[i] * numbers0[i] ) != (numbers1[i]  * numbers1[i] )) then
9.              if (numbers0[i]  != 0 && numbers1[i]  != 0) then
10.                 equally := false
11.             end
12.         end
13.     end
14.   end
15. end
```

```
int numbers1[4] = { 0,1,2,3 };
int numbers0[4] = { 0,1,2,3 };
```

equally	numbers0[i]	numbers1[i]	i	Überprüfung	Befehl
1	-	-	-	numbers0.size() != numbers1.size()	-
1	-	-	-	i >= 0 && equally == true	FOR(i = numbers0.size() -1; ; i--)
1	3	3	3	true	-
1	3	3	3	numbers0[i]^2 != numbers1[i]^2	-
1	3	3	3	i >= 0 && equally == true	-
1	2	2	2	true	-
1	2	2	2	numbers0[i]^2 != numbers1[i]^2	-
1	2	2	2	i >= 0 && equally == true	-
1	1	1	1	true	-
1	1	1	1	numbers0[i]^2 != numbers1[i]^2	-
1	1	1	1	i >= 0 && equally == true	-
1	0	0	0	true	-
1	0	0	0	numbers0[i]^2 != numbers1[i]^2	-
1	0	0	0	i >= 0 && equally == true	-
1	-	-	-	-	END

```
int numbers1[4] = { 0,1,2,3 };
int numbers0[4] = { 1,1,1,1 };
```

equally	numbers0[i]	numbers1[i]	i	Überprüfung	Befehl
1	-	-	-	numbers0.size() != numbers1.size()	-
1	-	-	-	i >= 0 && equally == true	FOR(i = numbers0.size() -1; ; i--)
1	1	3	3	true	-
1	1	3	3	numbers0[i]^2 != numbers1[i]^2	-
1	1	3	3	true	-
1	1	3	3	numbers0 != 0 || numbers1 != 0	-
1	1	3	3	true	equally := false
0	1	3	3	i >= 0 && equally == true	-
0	-	-	-	-	END

```
int numbers1[4] = { 0,1,2,3 };
int numbers0[1] = { 1 };
```

equally	numbers0[i]	numbers1[i]	i	Überprüfung	Befehl
1	-	-	-	numbers0.size() != numbers1.size()	-
1	-	-	-	true	equally := false
0	-	-	-	-	END

b)