Programming Concepts

Exercise 2

October 5, 2016

FH JOANNEUM
University of Applied Sciences

Internet Technologien & Anwendungen

Introduction

Bar codes are nearly ubiquitous in our modern world. Nearly every store bought product has one. On of the more common forms of bar code is the **EAN-8** bar codes. Sometimes bar codes become hard to read and must be entered manually or bar codes are wrongly read by the scanner.

In order to protect against input errors barcodes have a check digit which can be computed from the previous digits. If the scanned check digit does not match with the one that is computed a read error has occurred and the bar code must be read again.

What do do

Write a program that computes EAN-8 check digits. The user should enter 7 numbers as command line arguments and the program should print out the calculated check digit. If the users enters less than 7 numbers an error message should be printed.

```
$ ./exercise_2_yourname 5 5 1 2 3 4 5
The calculated check digit is: '7'.
```

Reading the digits

The digits are passed as command line arguments, Each digit itself is single argument. Before you can compute the check digits you must first transfer the digits from the `argv` into an integer array. You can convert the digits (which are strings) into integers by using the `atoi` function. To use this function you must first include `stdlib.h`.

How the check digit is calculated

The check digit is very easy to calculate. Every **odd** number of the check digit is multiplied by **3** and every **even** number is multiplied by 1 (i.e. it stays the same). Use this rule to build a sum over the bar codes 7 digits. Calculate the difference between the previously calculated sum and the next largest number that is **divisible by 10**. This number is your check digit!

Hint: Use the modulo (%) operator to calculate the check digit of the sum.

Barcode	5	5	1	2	3	4	5
Multiplier	3	1	3	1	3	1	3
Value	15	5	3	2	9	4	15
Sum:	**53**						
Check Digit:	**60 - 53 = 7**						

Table 1: Example of a check digit calculation

You can use the GS-1 check digit calculator at http://www.gs1.org/barcodes/support/check_digit_calculator to test against your implementation.

Non functional requirements

The program should consist out of multiple source files.

- `main.c` contains the main function and should convert the input from the command line into an integer array. If an error i.e. invalid input occurs the return value of the program should be non-zero.,
- `ean.c` should contain the function for calculating the check digit
- `ean.h` should contain the function prototypes for `ean.c` and also relevant constants.
- Use `#ifdef`s to output debug information about the check digit calculation when the preprocessor symbol `DEBUG` is set.

Submission

Submit the exercise on the E-Learning platform. Since this exercise will have multiple source files **you should zip** your exercises. Upload your assignment to the exercise folder with the name **Exercise 2**.

Please use the following naming convention: `exercise_2_<lastname>.zip`.

When you zip your solution take care that your source files are in the 'root' of your zip archive and not in a sub folder!