ENGR151 — Accelerated Introduction to Computer and Programming

Homework 2

Manuel — UM-JI (Fall 2021)

- MATLAB: write each exercise in a different file
- C/C++: use the provided assignment template
- Include simple comments in the code
- If applicable, split the code over several functions
- Extensively test your code and impove it
- Write a single README file per homework
- Carefully follow the git submission process

*Exercises preceded by a * are mandatory.*

JOJ Online Judge

Exercises 2, 4, 5 and 6 can be tested on JOJ Online Judge.

Important reminders regarding the Online Judge (OJ):

- For each exercise save all the files, without any folder structure, into a `.tar` archive;
- Strictly stick to the input and output formats provided in the specifications;
- The OJ only checks the correctness of the code not its quality;
- For feedbacks on the quality, submit the code as part of the assignment and include the OJ score as well as the failed cases in the README file;

Ex. 1 — *Algorithms and basic loops*

In the Gregorian calendar a regular year is composed of 365 days, and a "leap year" of 366 days, the 29th of February being added in order to reflect more precisely the orbital period of the Earth around the Sun. Determining whether a year is a leap year or not in the Gregorian calendar is done as follows: if a year is not divisible by 4 it's a regular year; if it's divisible by 100 then it's not a leap year, unless it's also divisible by 400.

Write a MATLAB script returning whether a given year is a leap year or not. The user should be prompted for a year until he enters a valid one.

* Ex. 2 — *Algorithms and loops*

A Pythagorean prime is an odd prime number that can be written as the sum of two squares. Such primes are of the form $p = 4n + 1$, for some integer n. Write a MATLAB script that (i) reads a number from the keyboard, then (ii) finds the next Pythagorean prime and (iii) returns the two corresponding squares.

Specifications.

- Input format: one line containing a number x
- Output format: one line containing a string of the form `p = a^2 + b^2`, where p is a prime, a and b are integers, and such that $p > x$ and $a \leq b$

Ex. 3 — *Recursion*

Chapter 3 describes an algorithm to read numbers digit by digit (slide 3.22). Instead we now want to read them as proper numbers, e.g. 452 should be "four hundred and fifty two". Write some MATLAB function and subfunctions such that an integer input less than 999,999,999 is returned as words.

Output rules.

- Do not print hyphens;
- Write "one hundred", not "a hundred";
- Add "and" after every hundred unit, even when it is zero, but unless it is followed by two zeros; e.g. 200 → two hundred, 221 → two hundred *and* twenty one, 1002 → one thousand *and* two;

* **Ex. 4 —** *Mathematical functions, loops, and recursion*

Given a continuous function f over an interval $[x_0, x_1]$ such that $\text{sign}(f(x_0)) \neq \text{sign}(f(x_1))$ find $r \in [x_0; x_1]$ such that $f(r) = 0$. The secant method is defined through the following recurrence relation

$$x_n = \frac{x_{n-2}f(x_{n-1}) - x_{n-1}f(x_{n-2})}{f(x_{n-1}) - f(x_{n-2})}$$

Write two MATLAB functions: one iterative and one recursive. Their inputs should be (i) a mathematical function, (ii) an interval containing a root, and (iii) a precision (number of decimal places). They should return the root of the function in the interval provided in the input.

Specifications.

- Input format: three lines containing (i) a function handle (e.g. `@(x) x^2-3*x+1`), (ii) an interval (e.g. `[ 0 1 ]` containing a root of the function on the previous line, and (iii) an integer representing the precision (e.g. 3)
- Output format: a single line showing the result

* **Ex. 5 —** *Control statements*

A positive integer n is an Armstrong number if the sum of the i-th power of each of its digits is n itself, with i the number of digits in n. For instance 153 is an Armstrong number for (i) it has three digits and (ii) $1^3 + 5^3 + 3^3 = 153$. Similarly 1 is also an Armstrong number since (i) 1 has 1 digit and (ii) $1 = 1^1$. Write a MATLAB function which given a number n returns the next Armstrong number or n if n is an Armstrong number.

Specifications.

- Input format: a single line expecting an integer n
- Output format: a single line displaying the first Armstrong number larger or equal to n

* **Ex. 6 —** *Basics on functions*

Given a date Zeller's Congruence formula allows to determine the corresponding day of the week. The

formula is as follows:

$$day = 1 + \left(d + \left\lfloor \frac{13m - 1}{5} \right\rfloor + y + \left\lfloor \frac{y}{4} \right\rfloor + \left\lfloor \frac{c}{4} \right\rfloor - 2c \right) \mod 7,$$

where d is the day of the month (1–31), m the number of the month (from March=1 to February=12), y the year of the century (14 for 2014) and c the century minus one (20 for 2014); the value *day* is an integer between 1 and 7, with 1 representing Sunday. Assign January and February to previous year.

Write a MATLAB function which takes as input a date in the format `[Day Month Year]` in the usual human format, and returns the corresponding day of the week (e.g. on the input [19 1 2012] the function should return Thursday).

Specifications.

- Input format: a single line containing a vector representing the date (e.g. `[ 19 1 2012 ]`)
- Output format: a single line containing a day of the week written as a string (e.g. `Thursday`)