VG101 — Introduction to Computer and Programming

Assignment 5

Manuel — UM-JI (Fall 2018)

- 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 assignment
- Archive the files (`*.{zip|tar}`) and upload on Canvas

*Exercises preceded by a * are mandatory. Any student not completing all of them, or submitting a work that cannot compile or be interpreted will automatically be deducted 1 mark on the final course grade.*

JOJ Online Judge

Exercises 1 to 4 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 — *Array*

Write a program taking an integer n as input and displaying all the primes less the n. All the primes should be stored in an array.

Specifications.

- Input format: one line with one integer n (e.g. 6)
- Output format: a space separated list of primes on one line (e.g. 2 3 5)
- Use comments to clearly indicate which array contains the primes

Ex. 2 — *Arrays and functions*

Write a function which takes as input a month, and the name of the first day of the month. It should display the calendar for the requested month.

Specifications.

- Input format: one line with one integer and the three first letters of a day (e.g. 6 Fri)
- Output format: similar to the sample output below

Sample output (ex. 2)

```
June
 Sun    Mon   Tue   Wed   Thu   Fri   Sat
                                  1     2
   3     4     5     6     7     8     9
  10    11    12    13    14    15    16
  17    18    19    20    21    22    23
  24    25    26    27    28    29    30
```

* **Ex. 3 —** *Strings*

Write a program to find the number of times a given string occurs in a sentence. The user will input both the sentence and the word.

Specifications.

- Input format: two lines, the sentence on the first and the word on the second
- Output format: one line showing the result

Sample output (ex. 3)

```
$ ./h5 -ex3
Input a sentence: good morning, have you seen the cat and the dog?
Input a string: the
The string 'the' occurs 2 times
```

Ex. 4 — *Loops, standard library, mathematical functions*

A projectile is fired into the air at an initial speed $v_0 = 30$ m.s^{-1} and an initial angle $\theta_0 = 30°$, with an initial height of 1.5m. Its trajectory can be expressed as

$$y = 1.5 + \tan(\theta_0)x - \left(\frac{g}{2v_o^2 \cos^2 \theta_0}\right) x^2,$$

where $g = 9.81$ is the gravitational acceleration constant.

Implement the bisection method to determine when the projectile touches the ground. Use the constant FLT_EPSILON as the convergence tolerance and $[75, 85]$ as the initial interval for the bisection method.

Hint 1: `FLT_EPSILON` is defined in `<float.h>` file and the mathematical functions are defined in `<math.h>`.

Hint 2: the trigonometric functions expect angles in radians

Specifications.

- Input format: one line with a space separated list in the order v_0 θ_0 height range_min range_max (e.g. 30 30 1.5 75 85)
- Output format: one line showing the result with six decimals (e.g. 81.970131)

* **Ex. 5 —** *Loop, array, and sorting*

The goal of this exercise is to write a C program that simulates a deck of 52 cards. Start by printing the cards in the following order, $2 < 3 < ... < 10 <$ *Jack* $<$ *Queen* $<$ *King* $<$ *Ace* assuming *Spades* $<$ *Hearts* $<$ *Diamonds* $<$ *Clubs*. Then shuffle them, print them out in their shuffled order, sort them following the above order, and print the resulting deck.

Specifications.

- A total of three decks should be displayed
- The user should press enter each time a deck is printed

Pair programming

The goal of the following exercise is to practice programming as *a pair*. For a better group work experience the following scenario is recommended.

- Sit in a comfortable environment and work together as a team;
- A student plays the "Driver" and the other one the "Navigator";
- The *driver*'s work is to type on the keyboard while the *navigator* provides suggestions;
- Both the *driver* and the *navigator* should pay attention to common typos and errors;
- Roles can be exchanged after a while;
- Both students are expected to think of the whole problem;

* **Ex. 6 —** *Low level C programming*

The following program performs a multiplication using an algorithm similar to the one from Karatsuba.

1. Detail Karatsuba algorithm in the README file (search it on internet).
2. Add comments to the code to describe what is done, line by line.
3. Explain in the README file what specific adjustments were made to the algorithm in order to improve the efficiency.
4. Search online what is a divide an conquer strategy.
5. Using a divide and conquer approach, together with the operators `&, |, << and >>` , write an efficient function to replace the for loops marked as "not optimal".

Low level multiplication (ex. 6)

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define SWAP(a,b) { a ^= b; b ^= a; a ^= b; }
unsigned long int mult(unsigned long int a, unsigned long int b);
int main () {
    unsigned long int a, b;
    srand(time(NULL));
#ifndef TEST
    a=rand(); b=rand();
```

```
    printf("%ld*%ld=%ld %ld\n",a,b,mult(a,b), RAND_MAX);
#endif
#ifdef TEST
    int i;
    for(i=0; i< 1000000; i++) {
        a=rand(); b=rand();
        if(mult(a,b)!=a*b) {
            fprintf(stderr,"Error (%d): a=%ld, b=%ld, a*b=%ld, k(a,b)=%ld\n",\
                i,a,b,a*b,mult(a,b));
            exit(-1);
        }
    }
#endif
}
unsigned long int mult(unsigned long int a, unsigned long int b) {
    int i, n, N;
    unsigned long int x0,y0,z0,z1=1;
    if(a<b) SWAP(a,b);
    if(b==0) return 0;
    for(n=-1, i = 1; i <= b; i<<=1, n++); /* not optimal */
    for(N=n; i <= a; i<<=1, N++);

    y0=b&((1<<n)-1);
    x0=a&((1<<N)-1);
    z0=mult(x0,y0);
    i=N+n;
    return ((z1<<i)+(x0<<n)+(y0<<N)+z0);
}
```