Self-Verification Test

This computer-based test requires the following equipment:

- C compiler
- Matlab

The test has a duration of three hours. It consists of three parts: Understanding and writing C code, as well as Matlab programming. The maximum score you can get is 100 points; while this verification test is ungraded, you will need to save your answers. Once all participants have completed the test, we will release the problem solutions which you can use to calculate your score. You will then need to upload your score in an anonymous survey on moodle.

The test is open book, i.e. all printed/written material is allowed (e.g., books, lecture notes, exercises, solutions, personal notes). You should use a computer from the computer room running Ubuntu. Note that we will not provide support for any installations on personal computers. **In parts II and III only** you can use internet to look for necessary information, but **do not communicate with other students. Additionally, as this is a self-assessment test, refrain from using chat-GPT and similar tools to answer the questions.**

Getting Started

To start with this test, you will need to download the material available on Moodle. Download `self-test-material` and extract it in your home directory. The uncompressed folder has the following contents:

- *part2/rnd_print, part2/fibonacci, part2/struct_copy* and *part2/count:* files needed for Part II.
- *part3/:* Matlab files needed to answer Part III.

Terminal Commands Cheatsheet

Changing directories: `cd directory`

List files in directory: `ls`

Running executable: `./executable`

Compile code with gcc: `gcc files -o executable`

Compile with makefile: `make`

Remove executables from directory: `make clean`

Part I: Understanding C Programming (32 points)

Do not use internet for this part.

1. (14pts): Code comprehension
   a. (8pts) What is the resulting output of the following code?
   b. (3pts): What is the difference between %d and %f (on line 22)?
   c. (3pts): What purpose does the return 0 (on line 23) serve?

```
1   #include <stdio.h>
2
3   float half = 1/2;
4   int one = 1;
5   int two = 2;
6
7   int func1(int a, int b){
8       int c = a + b;
9       c *= 2;
10      return c;
11  }
12
13  float func2(int a, float b){
14      return a * b;
15  }
16
17  int main(void){
18      int a = two + one;
19      float b = half + half + one;
20      int c = func1(a, (int) b);
21      float d = func2(a,b);
22      printf("%d %f %d %f \n", a, b, c, d);
23     return 0;
24  }
```

2. (8pts): Code comprehension
   a. (4pts) What is wrong with the following code snippet?
   b. (4pts) Even with this error, the program executes. What value will be printed?

```
#include <stdio.h>

int function(void){
        int x = 1;
        int *y = &x;
        int **z = &y;
        return *z;
}

int main (void){
        int value = function();
        printf("The value = %d \n", value);
        return 0;
}
```

3. (10pts): Building code:
    a. (2pts) What is the main difference between a compiled language (such as C) and an interpreted language (such as Python)?
    b. (4pts) There are two main steps involved in building an executable, compiling and linking. What does each of these steps do?
    c. (2pts) What is the role of the preprocessor in the compilation process?
    d. (2pts) What does it mean when a function is "declared" but not "defined"?

Part II: C Programming (38 points)

4. (10pts): Open the file rnd_print.c located in folder part2/rnd_print. Complete the main function to generate and print a random number between 1 and 100. Compile using make and test your function. Important: The number generated should be different every time you call the program. *(Hint: If needed you can also import functionality from other libraries. Useful functions you might look for could be rand(), srand(), time().)*

5. (5pts): Open the file *main.c* located in folder *part2/fibonacci*. This file contains the main function of a program intended to print the n (user input) first Fibonacci numbers. In the *fibonacci.c* file, implement a function `fibonacii()`, which calculates and prints the fibonacci numbers. You'll also have to add the function declaration to the .h file. Compile using make and test your function. (*Hint: Fibonacci number N is defined as: F(N) = F(N-1) + F(N-2).)*

6. (8pts): Open the file *struct_copy.c* located in folder *part2/struct_copy*. This program contains two types of structures, A and B, where B is a simplified version of A (without the "address" field). Structure A already has an initialized instance.
Write a function void struct_cpy(struct B* b, struct A a) that copies the common fields of A to B. Use the function struct_print(struct B b) to print the copied fields of structure B. Compile using make and test your function. (*Hint: you can use a function of the string.h library to copy strings.)*

7. (15pt): Open the file *count.c* located in folder *part2/count*. Write a program that takes an array of length L as user input and counts how many times numbers from 0 to 9 appear in it.
Input: The first line of the input should be the length L of the array. The following lines should be the elements of the array, which are decimal numbers from 0 to 9.
Output: The output of the program should print one line per number from 0 to 9. For each line, print the number and how many times it appears in the input array
Example: In the following example, the user provides an array of letters of length 6 and its content: `2 5 7 2 2 5`. The numbers that appear in this array are 2, 5 and 7 and they appear 3, 2 and 1 times, respectively. The program outputs one line for each number from 0 to 9 that prints the number and how many times it appears in the array.

Input	Output
➢ 6 // array length	➢ 0 0
➢ 2 // array content	➢ 1 0
➢ 5	➢ 2 3
➢ 7	➢ 3 0
➢ 2	➢ 4 0
➢ 2	➢ 5 2
➢ 5	➢ 6 0
	➢ 7 1
	➢ 8 0
	➢ 9 0

Hint: In the file count.c you fill find the declaration of the array `count_array` *and a function called* `count`*. Use* `count_array` *and the function* `count` *to store how many times a number appears. Understanding what is happening inside the function will help you to solve this problem.*

Part III: Matlab Programming (30 points)

8. (14pts) Open the file matrix_main.m located in folder part3/.
   a. (4pts) Create a function generate_random_matrix which returns a 5x5 matrix filled with random integers between 1 and 10.
   b. (2pts) In the main script, call the function created above to obtain matrix A and print it to the console line.
   c. (2pts) Set A(1,2) and A(3, 4) to zero
   d. (2pts) Calculate and print $B = A - A^T$, with A^T being the transpose of matrix A.
   e. (4pts) Calculate and print $C = A * \mathrm{A}^{\mathrm{T}}$, as well as $D = A \cdot A^T$. With * the standard matrix multiplication and · being element-wise multiplication.
9. (16pts) Plotting, open the file plotting.m located in folder part3/:
   f. (3pts) Generate a vector x that ranges from -2π to 2π with a step size of 0.1.
   g. (2pts) Calculate a corresponding vector "y" given by: $y = sin(x) + cos(2x)$.
   h. (3pts) Plot y as a function of x using a blue solid line.
   i. (2pts) Indicate the point (0,0) with a red O on the same plot.
   j. (2pts) Add a title to the plot, label the x and y-axes appropriately.
   k. (4pts) Find and print the maximum value of y and the corresponding value of x.