PART 1: Declaration of variables and printf()

1. Create a file `hello.c`. In this file write the following code:

```
#include <stdio.h>

int main()
{
      printf ("Hello World!\n");
      return 0;
}
```

Notes:

1. In order to compile and run your program using Visual Studio Code you can just use the option `Run Code` and see the results on the Output tab (or Terminal tab). If there is any compile error, you must fix it and `Run Code` again.
2. In order to compile your program using the linux environment, first you need to execute the compilation command and then, if the compilation hasn't produced any errors, execute the created executable file:

```
gcc  hello.c -o hola

./hola
```

3. The compilation process has translated your program into machine code, that's why you can later run it as many times as you want without recompiling it.
4. In this case, the `printf()` function shows in the screen the string (output message) passed as the argument. The string is the only compulsory parameter for this function.
5. In the string that we pass we have added the sequence \n. This sequence represents a line break. There are, obviously, more sequences, such as \t that represents a tabulation.

2. Modify the program by adding an integer variable declaration `x` with an initial value of 5. Print the value of the variable calling the `printf()` function. The final code ends up like this:

```
#include <stdio.h>

int main()
{
      int x = 5;
      printf( "Hello World! %d\n", x);
      return 0;
}
```

Compile the program and run it.

Notes:

1. Now, in the string being printed the %d sequence appears. It indicates that in this position of the output message an integer value is going to appear. This integer value is stored in a parameter passed to the function (`x`, in this case).
2. For values of different types, we use different sequences, such as %f for real values, %c for character values, or %s for string values.

3. Modify the program, adding a declaration for a variable (`y`) of type `float` initialized to 2.5, and a variable (`c`) of type `char` initialized to 'z'. Print the values of these variables by modifying the call to the `printf()` function. What is the final code of the program `hello.c` if the result appeared on the screen is as follows:

```
2.5

Hello World! 5, the character is: z :-)
```

PART 2: Functions and parameters

1. Create a file `functions.c` with the following code:

```
#include <stdio.h>
#include <stdlib.h>

int Max (int a, int b)
{
        if (a < b) return b;
        return a;
}

int main()
{
      int x, y, z;
      // Always generate the same pseudo-random sequence
      // of numbers
      srand(1234);
      x = rand () % 10000;
      y = rand () % 10000;
      z = Max (x, y);
      printf(??);
      return 0;
}
```

Complete the `printf()` call in order to provide the following output message with the generated numbers:

```
The maximum value between <x> and <y> is <z>"
```

2. Add a function to your program to compute and return the sum of all the divisors of a given integer. The function's declaration could be as follows:

```
int SumDiv (int num);
```

Add to `main()` the required variables and calls needed to compute the sum of the divisors of two numbers `x` and `y`. Add also the `printf()` function call that displays the results giving a following message:

The sum of the divisors of <x> is <result for x>.

The sum of the divisors of <y> is <result for y>.

3. Implement a function that calculates Fahrenheit to Celsius. This function takes as parameter a number (Fahrenheit temperature), changes this value to Celsius and returns the new value. In the main function, ask for the Fahrenheit temperature, invoke the implemented function passing the introduced temperature. Show the result on the screen.

PART 3: Arrays

1. Create a file `arrays.c`. In this file write the following program:

```
#include <stdio.h>
#include <stdlib.h>

#define N 100

void InitArray (float vect[N])
{
      int i;
      for(i=0; i<N; i++)
            vect[i] = (rand()%100)/100.0;
}

int main()
{
      float v1[N], v2[N];
      srand(1234);
      InitArray (v1);
      InitArray (v2);
      return 0;
}
```

2. Add to your program a function to search for and return the maximum value of an array. Modify the `main()` function so that this function is called twice to get the maximum value of the array `v1` and then `v2`. Add the necessary `printf()` to display the results obtained. The function's declaration could be as follows:

```
float MaxArray (float v[N]);
```

3. Add to your program a function to compute and return the sum of the values of the elements of an array. Modify the `main()` function again in order to get the sum of the elements of the vector `v1` and then `v2` and display the result on the screen. The function's declaration could be as follows:

```
float SumArray (float v[N]);
```

4. Add to your program a function to compute the sum of two vectors (which will be stored in a 3rd vector). Implement also a function to display the first 5 elements of a vector on the screen. Modify the main function so that the sum of `v1` and `v2` is calculated, storing the result in `v3` (using the adding function you will implement), and display the first 5 values of `v1`, `v2`, and `v3` (to check that the result is correct).

   The functions' declaration could be as follows:

```
void AddVectors (float op1[N], float op2[N], float res[N]);
void PrintArray (float v[N]);
```

PART 4: Exercises for practicing

Exercise 1 (basic level)

Write a function that sorts the three numbers passed as parameters in ascending order. The function must return the sorted values in ascending order. Then write a program that reads three numbers, calls the above function to sort them, and finally, prints the result to the screen.

Exercise 2 (basic level)

A company divides its customer base into three categories: Basic, Standard and Premium. A different discount is applied to each type of customer every time they purchase a product. The following criteria are applied:

- For "basic" customers -> no discount is ever applied.
- For "standard" customers -> a 5% discount is applied if the purchase amount is greater than 10,000€.
- Premium customers -> they receive a 20% discount if the purchase amount is over 10,000€, 10% if it is over 1,000€ and 5% if it is over 100€. If it is lower, no discount is applied.

Write a program that, first, requests the purchase amount and the customer type (as a character: 'B' for basic customers, 'S' for standard customers and 'P' for premium customers). Then, the program displays the final amount to be paid by the customer according to the conditions explained before. Note, that you should check that the value

entered for the customer type is one of the valid options ('B', 'S', 'P'). If it isn't, instead of showing the final amount on the screen, the message "ERROR" should be displayed.

Exercise 3 (basic level)

Write a program that, given the number of operations N that we want to compute, displays the series of all possible sums from 1 up to N (for each i from 1 to i<=N), in the indicated format. For example, if N = 3, the output of the program should be:

```
1=1
1+2=3
1+2+3=6
```

Exercise 4 (advanced level)

Write a program that, for a number N greater than 2, displays on the screen the first N numbers of the Fibonacci sequence. Remember that the Fibonacci series is defined as follows:

F(0)=0
F(1)=1
F(2)=1
F(n)=F(n-2)+F(n-1)

The output format should be as follows (assuming, for example, N=7):

```
Fibonacci sequence for N=7:
F(0)=0
F(1)=1
F(2)=1
F(3)=2
F(4)=3
F(5)=5
F(6)=8
F(7)=13
```

Exercici 5 (advanced level)

Write a program that computes all perfect numbers between 1 and a given number N. A perfect number is one that the sum of all its divisors is the number itself. For example, the divisors of 6 are 1, 2 and 3, and (1 + 2 + 3) = 6. The program must call the `isPerfect()` function, which receives a number as a parameter and returns a boolean value, 1 if it's true (a given number is perfect) or 0 if it's false (a given number is not perfect). Note that in C there is no boolean type.

The output of the program should be as follows (assuming, for example, N=500):

```
Enter a number:
The perfect numbers for 500 are: 6, 28, 496
```