Lab Sheet No 1. Subprograms (Functions and Procedures)

1) Objectives

The objective of this lab is to learn, through a set of activities and practical exercises, the various types of subprograms and their use in the C language.

By the end of this lab, the student should be familiar with modular programming and be able to declare, call, and use subprograms correctly in the C language.

2) Example 1: calculation of the value of $f(x)$

The following program allows to calculate $f(x)$ for two different values of x, where: $f(x) = x^2 - 2x + 1$.

The calculation of $f(x)$ is performed in a separate function.

```
#include <stdio.h>
int calculate_f(int a){
   int f;
   f = a * a - 2 * a + 1;
   return f;
}
int main() {
   int x, y;
   x = 2;
   y = calculate_f(x);
   printf("f(%d) = %d\n",x, y);
   x = -4;
   y = calculate_f(x);
   printf("f(%d) = %d\n",x, y);
   return 0;
}
```

1. Create a new project and type the above code.
2. Modify the program to calculate $f(x)$ for multiple values of x entered by the user. The number of these values should also be entered by the user.
3. Make the necessary modifications so that the calculation of $f(x)$ is performed in a procedure instead of a function.

3) Application Exercises

1. Given A and B, two natural numbers, we want to establish a program that obtains a number C equal to the concatenation of the two numbers A and B.

   For example, if $A = 276$ and $B = 15$, the result of concatenation would be the number $C = 27615$.

   However, the number C can be obtained using the following formula:

   $$C = A \times 10^{nb} + B$$

   Where nb is the number of digits in B.

   To solve this problem, write the following functions:

   a) A function `countDigits(n)` that counts and returns the number of digits in a number `n` passed as a parameter.

   b) A function `concatenate(A, B)` that calculates and returns the number resulting from the concatenation of two numbers `A` and `B` passed as parameters. Concatenation is done using the formula described above.

      You can use the predefined `pow` function to calculate the power.

   c) The `main` function that allows the user to enter two natural numbers, concatenate them, and display the result.

2. We want to calculate the number of combinations of P elements taken from N elements, knowing that this number is determined by the following formula:

   $$C_N^P = \frac{N!}{P! \times (N-P)!}$$

   a) Write the function `factorial` that returns the factorial of an integer passed as a parameter.

   b) Write the function `combination` that takes two numbers N and P as input and calculates and returns the combination of P elements among N.

   c) Write the `main` function that prompts the user to enter two numbers N and P, calculates, and displays the number of combinations of P elements taken from N elements by calling the previous `combination` function.

3. Write a program to input an array of 10 real elements, reduce it, and display the result. The reduction involves subtracting the standard deviation of the array from each element. The standard deviation (σ) is given by:

   $$\sigma = \sqrt{\frac{1}{n}\left(\sum_{i=0}^{n-1}\left(T[i]-m\right)^2\right)}$$

   Where m is the mean of the elements in the array.

   The program should be decomposed into the following subprograms:

- Procedure **`fillArray`** to input values from the keyboard into an array passed as a parameter.
- Procedure **`displayArray`** to display the elements of an array passed as a parameter.
- Function **`calculateMean`** that calculates and returns the mean of the elements in an array.
- Function **`calculateStd`** which takes an array as a parameter and calculates and returns its standard deviation.
- Procedure **`reduceArray`** to reduce an array passed as a parameter.
- The **`main`** function which only calls the aforementioned subprograms.

4. Consider the sequence U_n defined by: $U_0 = 0$, $U_1 = 1$, $U_n = 3U_{n-1} - U_{n-2} + 5$
   a) Write a recursive function that takes an integer **`n`** as input and returns the **`n`**th term of the sequence U_n.
   b) Test this function in a main program.
5. Let **`T`** be an array of 6 integer elements:
   a) Write a recursive procedure **`sum_Max(T, i, s, max)`** that calculates and returns the sum and the maximum of the elements in an array **`T`**.
   b) Write the **`main`** function that fills the array **`T`** and calculates, then displays the sum and the maximum of its elements. The calculation of the sum and the maximum should be done using the previous **`sum_Max`** procedure.

4) Additional Exercises

6. A perfect number is a number that has the special property of being equal to the sum of all its divisors, excluding itself. For example, 28 is a perfect number because the proper divisors of 28 are 1, 2, 4, 7, and 14, and in addition, 28 = 1 + 2 + 4 + 7 + 14.
   a) Write the function **`isDivisor`** that takes two numbers **`a`** and **`b`** as input and returns **`True`** if **`b`** is a divisor of **`a`** and returns **`False`** otherwise.
   b) Write the procedure **`sumDivisors`** that calculates and returns the sum of divisors of an integer passed as a parameter.
   c) Write the function **`isPerfect`** which, given an integer **`n`**, returns whether it is a perfect number or not.
   d) Write the procedure **`displayAllPerfect`** which takes an integer **`n`** as input and displays all perfect numbers between 1 and **`n`**.
   e) Write the **`main`** function that displays all perfect numbers between 1 and 1000 by calling the **`displayAllPerfect`** procedure.

7. Eight football teams decide to have a tournament among themselves. During this tournament, each team carries the following information: Team name, City to which the team belongs, Number of points, Ranking

   a) Define a data structure **`Team`** to store the information of a team.

   b) Write a function **`inputTeam`** that reads all the information of a team.

   c) Write a procedure **`inputAllTeams`** that inputs the information of the eight teams and stores them in an array passed as an argument.

   d) Write a function **`maxPoints`** that returns the team with the highest number of points.

   e) Write a function **`searchTeam`** that takes a team name as a parameter and returns whether this team is part of the tournament.

   f) Test the above subprograms in a main program.

8. The Olympic average of a set of numbers is defined as the average of all numbers except the smallest and the largest. For example, for the numbers 2, 3, 13, 7, 8, the Olympic average is 6.

   We want to calculate the Olympic average of a set of 6 numbers stored in an array.

   a) Write a function **`sum`** that takes an array **`T`** as input and calculates and returns the sum of its elements.

   b) Write the procedure **`MinMax`** that returns the smallest and largest elements in an array passed as an argument.

   c) Write the function **`OlympicAverage`** that takes an array of integers and returns the Olympic average of its elements.

   d) Test the above function in a main program.

9. The Syracuse sequence of an integer a is defined recursively as follows:

   $u_0 = a$, and for any integer $n \geq 1$, we have :

$$u_{n+1} = \begin{cases} \frac{u_n}{2}, & If\ u_n\ is\ even \\ 3u_n + 1, & If\ u_n\ is\ odd \end{cases}$$

   a) Write a function **`isEven`** that takes an integer **`x`** as an argument and returns **`True`** if **`x`** is even and **`False`** if **`x`** is odd.

   b) Use the **`isEven`** function to write the recursive function **`Syracuse`** that recursively calculates the **`k`**th term of this sequence.