SCHOOL
SUPERIOR
MEDIA
ARTS
AND DESIGN
POLYTECHNIC
FROM PORT

AL P

EXERCISE SHEET

TEACHING ACTIVITY

Algorithms and Data Structures

CURRICULAR UNIT

Sheet 02 – Iterative Structures

RECORD

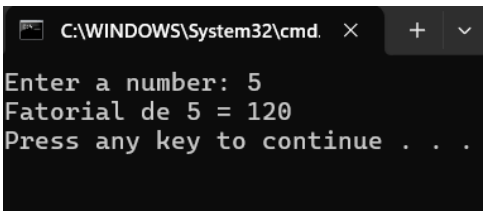
Level 1 (for loops)

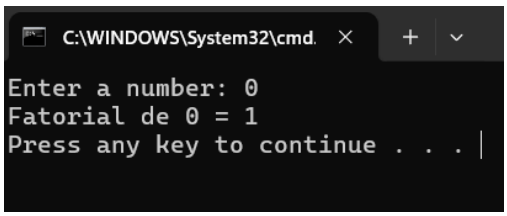
Basic

1. Develop a program that simulates the factorial function, that is, that determines the factorial of a given number.

   Example: Factorial of 5 = 5 * 4 * 3 * 2 * 1 = 120
   Note that 0! = 1

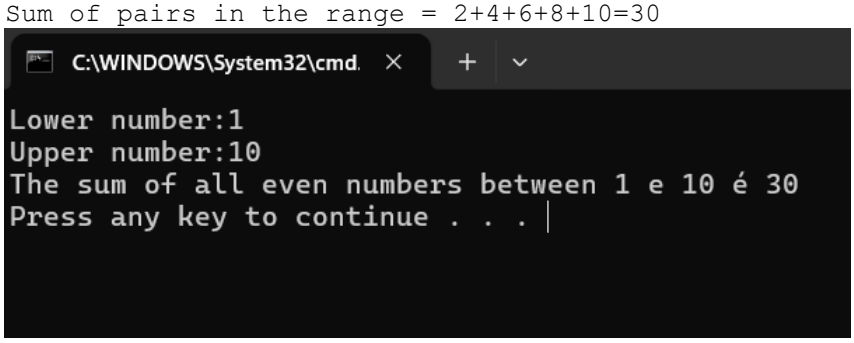
   Note: Do not use the ***math.factorial () function***
   The goal is to develop our own factorial function.





2. Implement a program that asks the user to indicate 2 integers (lower limit and upper limit), then calculates the sum of all pairs between that range (including the indicated limits).

   Example:
   Lower limit: 1
   Upper limit: 10
   Sum of pairs in the range = 2+4+6+8+10=30



▪ Level 2 (while loops)

Intermediate

3. Guess the number game!
   Develop a program that simulates the game of guessing a number.

   The program should start by generating a random number (between 1 and 50), allowing the player to iteratively try to guess the number generated by the computer.

Tip!

random library
(*import random*).
In this library you will find two functions to generate random numbers:

- **RANDOM.RANDRANGE (LININF,LIMSUP)**
- **RANDOM.RANDINT (LIMINF,LIMSUP)**

```
import random

num= random.randrange(0,10)  # Return random integer in range [a, b[, excludes the end points
num= random.randint(0,10)    # Return random integer in range [a, b], including both end points
```

The player has several attempts to guess the number, and after each attempt a message like this should appear:

- " **Number is bigger** " - if the player's guess is lower than the number to guess
- " **Number is smaller** " - if the player's guess is higher than the number to guess
- " **You got it!!!**" - if the player's guess matches the number to guess.

Other considerations:

• After 10 failed attempts the game should end, indicating the player's failure, with a message like "**You have used up all 10 attempts :(** ".

• When the player gets the number right, the game should indicate the number of attempts the player needed to get it right.

See the example bellow:

```
C:\WINDOWS\System32\cmd.

                    Guess the Number Game!

Enter your 1° guess: 25
The number is BIGGER

Enter your 2° guess: 35
The number is SMALLER

Enter your 3° guess: 30
The number is BIGGER

Enter your 4° guess: 33
The number is SMALLER

Enter your 5° guess: 32
The number is SMALLER

Enter your 6° guess: 31
Congratulations! You guessed the number in 6 attempts!
Press any key to continue . . .
```

4. Make a 2.0 version of the previous game where:

   After completing a game, the user should be given the option to start a new game: " **New game (Y/N)?**".
   The program must behave according to the response given by the user (**Y** or **N**).

- **Level 3 (for | while loops)** Advanced

5. Write a program that reads a number (integer and positive) and indicates whether it is a **prime number** or not.

   Note: A prime number is divisible **only** by itself and 1.

Números Primos entre 1 e 1000

Entre 1 e 1000 há 168 números primos, são eles:

2	3	5	7	11	13	17	19	23	29	31	37	41	43
47	53	59	61	67	71	73	79	83	89	97	101	103	107
109	113	127	131	137	139	149	151	157	163	167	173	179	181
191	193	197	199	211	223	227	229	233	239	241	251	257	263
269	271	277	281	283	293	307	311	313	317	331	337	347	349
353	359	367	373	379	383	389	397	401	409	419	421	431	433
439	443	449	457	461	463	467	479	487	491	499	503	509	521
523	541	547	557	563	569	571	577	587	593	599	601	607	613
617	619	631	641	643	647	653	659	661	673	677	683	691	701
709	719	727	733	739	743	751	757	761	769	773	787	797	809
811	821	823	827	829	839	853	857	859	863	877	881	883	887
907	911	919	929	937	941	947	953	967	971	977	983	991	997

```
C:\WINDOWS\System32\cmd.
Number:21
The number 21 IS NOT a prime number
Press any key to continue . . .
```

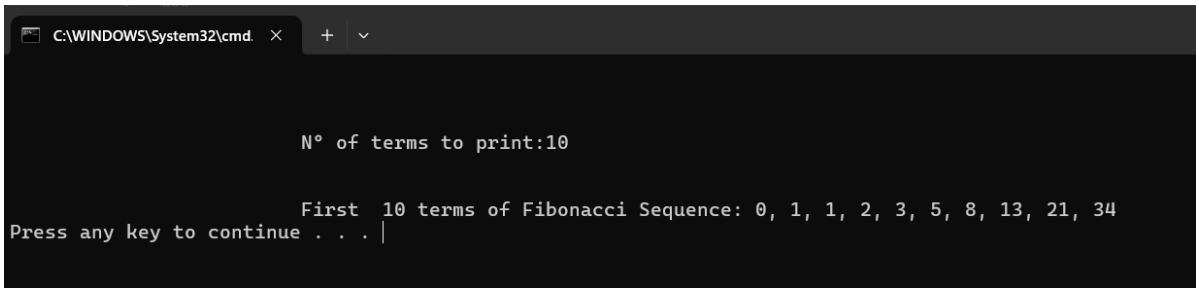
```
C:\WINDOWS\System32\cmd.
Number:13
The number 13 is a prime number
Press any key to continue . . .
```

6. Develop a program that illustrates the first **n** terms of the **Fibonacci sequence**, where the number of desired terms (**n**) must be indicated by the user.

   Fibonacci sequence , each term results from the sum of the two previous ones.
   Source: http://pt.wikipedia.org/wiki/N%C3%BAmero_de_Fibonacci

   > Os **números de Fibonacci** são, portanto, os números que compõem a seguinte sequência (sequência A000045 na OEIS):
   >
   > 0,1, **1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, ...** .[nota 1][2].

   Example:



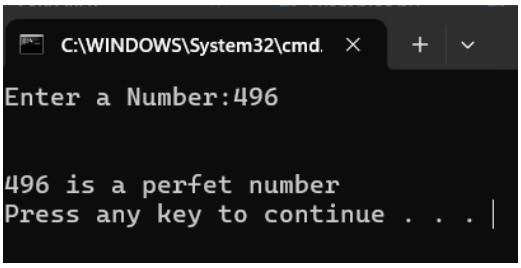
7. Write a program that checks if a given number (integer and positive) is perfect.

   In mathematics, a perfect number is an integer for which the sum of all its proper positive divisors is equal to the number itself.

   For example, the number 6 is a perfect number because:

   6 is divisible by: 1, 2 and 3. And 1+2+3 = 6, therefore it is a perfect number

   > Os quatro primeiros números perfeitos são:
   >
   > ✓ $6 = 1 + 2 + 3$
   > ✓ $28 = 1 + 2 + 4 + 7 + 14$
   > ✓ $496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248$
   > ✓ $8128 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 127 + 254 + 508 + 1016 + 2032 + 4064$

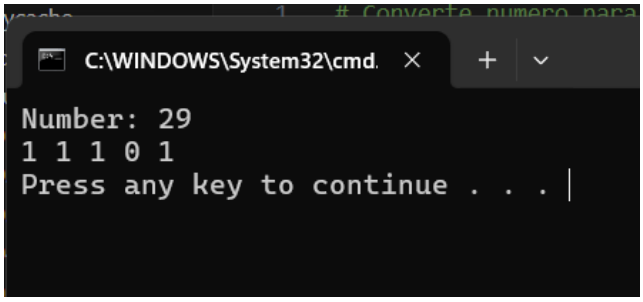


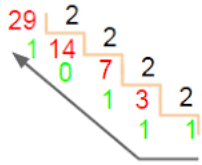
8. Implement a program that reads a number (between 1 and 99) and determines its representation in binary language.

   Example:

   Number: 12 Result: 1 1 0 0

   Number: 29 Result: 1 1 1 0 1





29 Decimal = 11101 Binário

9. Read a set of n integers (where n is previously specified by the user. Then determine the second largest value among the set of numbers read.

   Note: do not use arrays /lists to solve the exercise!