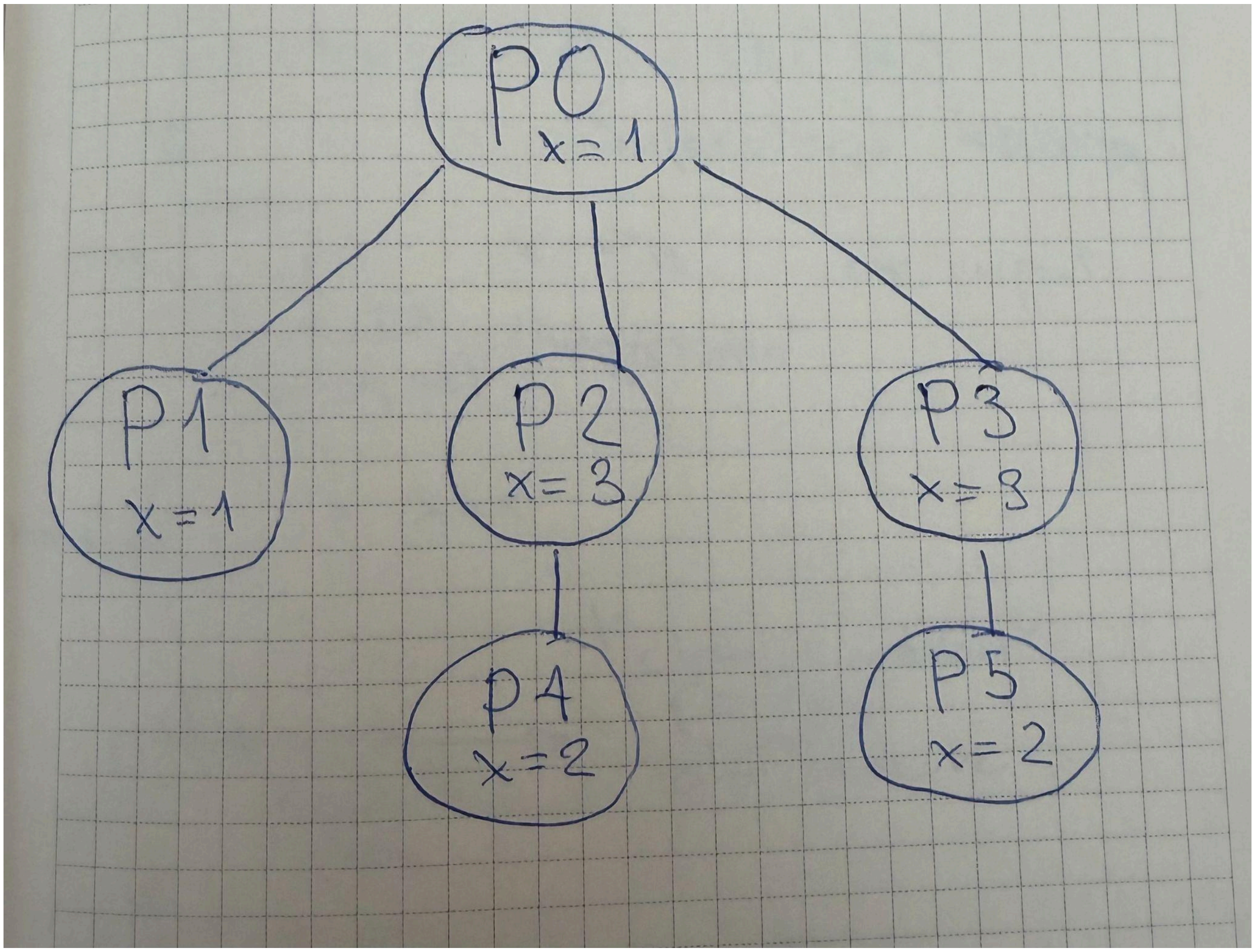
ASSIGNMENT #3

Problem 3.1: process creation using fork()

a) The program creates 6 processes during its execution

   x is initialized to 0

   1) At line 10: x++ increments so x = 1 in P0, then this value is inherited by all children processes
   2) After fork() in line 11 another P1 is created at x = 1
   3) In Line 12 both P0 and P1 do a fork so P2 and P3 children are created with x =1, but P0 and P1 don't enter the if block so their x stays at 1.
   4) Inside the if block on line 13:
      a) P2 and P3 fork so P4 and P5 are created
      b) P2 and P3 increment (x = 2), but P4, P5 stay at x = 1
      c) Then every process (P2,P3,P4,P5) increments so (P2,P3 | x = 3) and (P4,P5 | x = 2) – line 16



b) Program output:

```
p0: x = 1
p1: x = 1
p2: x = 3
p3: x = 3
p4: x = 2
p5: x = 2
```

Each process will be of the form p<offset>: x = <value>

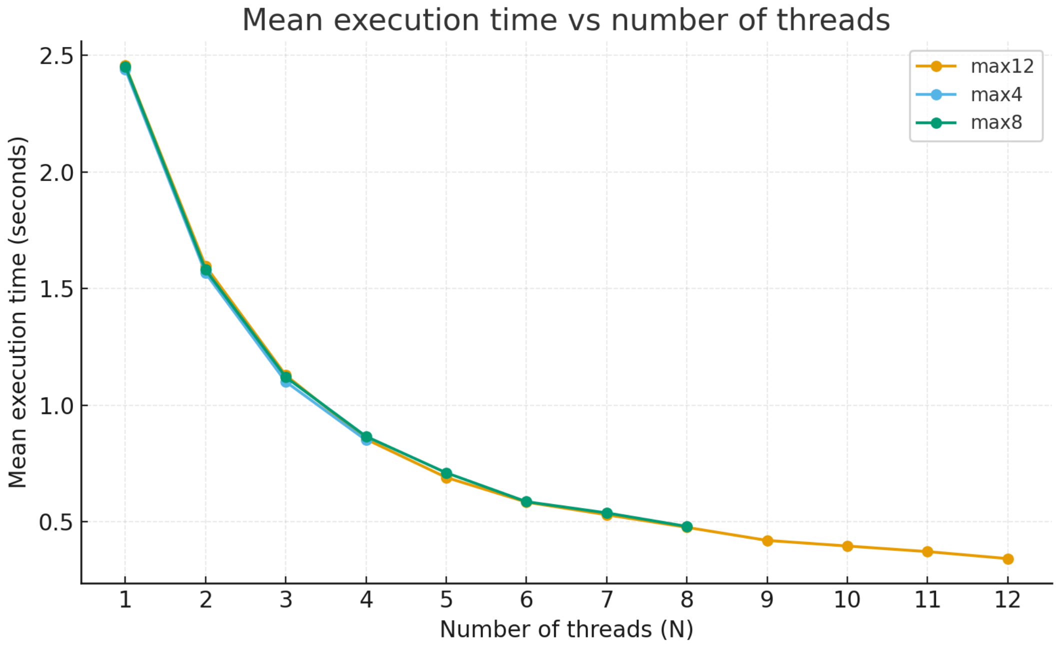
Problem 3.3: perfect threading

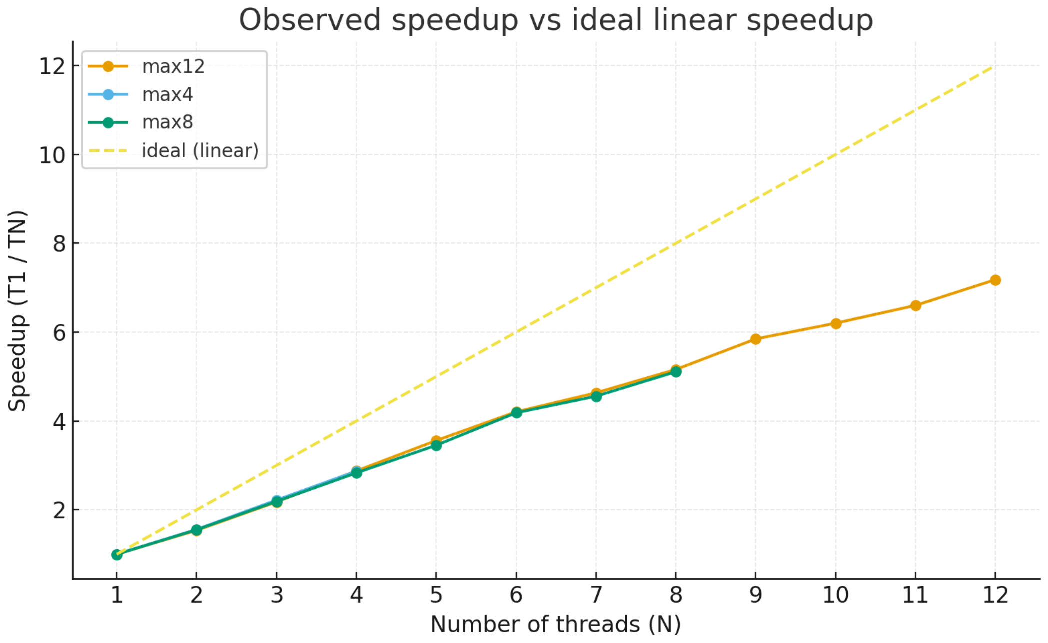
Part c)

I ran the script with these measurements:

Start = 1
End = 500000
Runs = 5
And used a different number of threads from 4, 8, 12

These are the plots I made from the results:





Speed summary from the runs:

4 threads: mean ≈ **0.85 s** , **speedup ≈ 2.88** , **efficiency ≈ 0.72**

8 threads: mean ≈ **0.48 s** , **speedup ≈ 5.1** , **efficiency ≈ 0.64**

12 threads: mean ≈ **0.34 s** , **speedup ≈ 7.2** , **efficiency ≈ 0.60**

The number of threads strongly reduces the wall-clock time so for the absolute shortest wall-clock time use more threads.