

Problem Sheet #7

Problem 7.1: dynamic linking (project euler 14)

(2+2+3+3 = 10 points)

The [Project Euler 14](#) problem is described as follows:

The following iterative sequence is defined for the set of positive integers:

$$\begin{array}{ll} n \rightarrow n/2 & \text{if } n \text{ is even} \\ n \rightarrow 3n + 1 & \text{if } n \text{ is odd} \end{array}$$

Using the rule above and starting with 13, we generate the following sequence: $13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

It can be seen that this sequence (starting at 13 and finishing at 1) contains 10 terms. Although it has not been proved yet (Collatz Problem), it is thought that all starting numbers finish at 1.

Which starting number, under one million, produces the longest chain?

NOTE: Once the chain starts the terms are allowed to go above one million.

The following Python script implements a naive solution for this problem.

Project Euler 14 written in plain Python

```
def collatz(n: int) -> int:
    i: int = 1
    while n != 1:
        if n % 2 == 0:
            n = n // 2
        else:
            n = 3 * n + 1
        i += 1
    return i

def main() -> None:
    lower: int = 1
    upper: int = 1000000
    longest: int = 0
    length: int = 0
    for n in range(lower, upper):
        l = collatz(n)
        if l > length:
            longest = n
            length = l
    print(f'longest collatz sequence in [{lower}:{upper}] is at {longest} (length {length})')

if __name__ == '__main__':
    main()
```

This solution is relatively slow and your job is to replace the Python function `collatz` with an equivalent function written in plain C by using dynamic linking.

- Write a header file `collatz.h` declaring your function `collatz()` and a C source file `collatz.c` implementing your function `collatz()`. Implement in a separate file a `main()` function solving the Project Euler 14 problem in C.

- b) Compile `collatz.c` into a shared library. Provide a suitable `Makefile` or `CMakeLists.txt` file.
- c) Familiarize yourself with the `ctypes` foreign function library of Python and in particular with the `ctypes.CDLL` class providing access to functions provided by a shared library in Python. Rewrite the Python code shown above to use the `collatz` function written in C.
- d) Measure the performance of the pure python implementation, your pure C implementation, and your python implementation using the `collatz()` function written in C.