POIS ASSIGNMENT 1

TASK 1

BUILD A PROVABLY SECURE PRG (PSEUDO-RANDOM NUMBER GENERATOR)

THEORY

Given an initial seed, a PRG produces a sequence of bit indistinguishable from a sequence produced by a real random source. Indistinguishable means that there is no algorithm executable in polynomial time on a probabilistic Turing machine that can decide if the given sequence is random or calculated.

Hence, here it is a definition of PRNG: it's an algorithm executable in polynomial time on a deterministic Turing Machine that calculates a function ***G*** such that

$$G : \{0,1\}^k \rightarrow \{0,1\}^{l(k)}$$

with ***l*** as a monotonically increasing function. That means the output is always longer than the input (seed).

It is possible to build any PRNG in the form of ***G*** as follows:

```
function G(x_0):
    x_1 . λ_1 = H(x_0)              // x_i is a kbit-string
    x_2 . λ_2 = H(x_1)              // λ_i is a single bit
    x_3 . λ_3 = H(x_2)
        ⋮
    x_l(k) . λ_l(k) = H(x_l(k)-1)
    return λ_1 . λ_2 . ... . λ_l(k)
```

where :

- $x_i \cdot \lambda_i$ is a bit string, result of the concatenation between the bit string x_i and the single bit λ_i.
- The ***H*** function generates a one bit longer sequence from the initial seed.
- By calling the ***H*** function ***l(k)*** times and taking just the last bit from each iteration, we have generated a sequence of ***l(k)*** bits. Obviously this function is ***G***.

We are now able to build a function that takes ***k*** bit and returns ***l(k)*** bit so that no algorithm can decide if the ***l(k)*** bit are generated from a real random source.

With this trick we moved from the problem to build a function that outputs ***l(k)*** bit (with ***l(·)*** generic polynomial function) to one that returns just ***k+1*** bits.