Lab-3. EE3401, Jan-Apr2024

We will use the python programming language for these exercises. If you don;t have python installed, use https://www.online-python.com/.

1A. Use python lists as a stack. It has pop and append which corresponds pop and push operations of a normal stack. Make a stack of integers to familiarize yourself.
B. Reverse polish notation (RPN): an expression like
2 + 3*4 can be written in RPN as 2 3 4 * +, so that the binary operands come before the operation itself. See https://www-stone.ch.cam.ac.uk/documentation/rrf/rpn.html. This sequence can be evaluated on a stack by following these rules:
a. If you encounter a number, push it on the stack.
b. If you encounter an operation, take the arguments from the top of the stack, apply the operand and store the result back on top of the stack.
Implement a function called “forth” in python which takes a RPN sequence and calculates its value using the stack made in part A.
C. Add following instructions to the function forth:

- DUP duplicates the top of stack (TOS)
- SWAP interchanges the top two elements
- DROP pops the TOS discarding it
- "." pops and prints the TOS
- "=" pops the top two elements, compares them, and push 1 onto the stack if there are equal, and 0 if no
- IF [instructions] THEN, check the TOS, if it is 0, then execute instructions that are listed till THEN.

Can you square or negate the TOS? Can you sum n numbers?

For help, see https://www.openbookproject.net/py4fun/forth/forth.html

2. Write a program for a machine to detect the sequence 0^i1^i, where i is a natural number. It has to check whether this string is in this format. The string is stored in an (infinite) list indexed by integers starting with 0. The machine can only read/write on the list a value x which can be {0, 1, B}.