Task 2: WAT-parser

------------------

For this task, you will write a lexer, parser and compiler that transforms WebAssembly Text Format (WAT) into WebAssembly bytecode (WASM).

The generated WASM of course has to be compliant with the WASM spec and be executable by any WASM engine, including your own VM from task 1!

Since there are several variants of WAT, we will focus on just one: the output of the wat-desugar tool in the wabt toolkit (https://github.com/WebAssembly/wabt).

MUST have features:

- Support all instructions and WASM features that you have to support for task 1

- ~~Support comments in both forms ( single-line ;; and inline (;...;) )~~

- Properly handle errors/unexpected syntax

- Proper error messages + graceful exit

- Problems in 1 function/section shouldn't mean other functions can't be properly compiled

Nice to have/expected for good score:

- ~~Support for $variable syntax~~

- The "constant folding" optimization (applied recursively)

- ~~Proper size indicators at the start of sections (so VM doesn't necessarily need to rely on FIXUPS)~~

- ~~Named func exports~~

Optional:

- Define a (partial) BNF for the WAT format and use a parser-generator to create the lexer/parser code

- You need to be able to also explain the parser-generator output, at least at a high level!

- Potential inspiration: https://janmidtgaard.dk/papers/Perenyi-Midtgaard%3AAPLAS20.pdf

- Potential inspiration: https://github.com/wasmerio/wapm-cli/blob/a11ea874e20c04fcba8bd33853cf6f908a5e4335/lib/wasm-interface/src/parser.rs

- Implement additional optimizations:

- Local variable elimination, function inlining, common subexpression elimination, etc.

Not required:

- Non wat-desugar WAT variants

- proper UTF-8 support for strings (you can assume all .wat input is ASCII)