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 for LEB128 encoded integer values and floats~~

- ~~Support for reading ASCII strings (e.g., export names, data segment contents)~~

- ~~Way to call WASM exported functions from the VM by string-name and at least print their results~~

- ~~"func" import (should be emulated/faked/hardlinked. No actual dynamic coupling with C++ functions necessary)~~

- ~~"data" section~~

- ~~"memory" import~~

- ~~"type" section~~

- Support for the following instructions

- ~~block, loop,~~ ~~if, else~~, ~~br, br\_if,~~ ~~return~~, ~~call~~

- ~~local.get, local.set, local.tee~~

- ~~i32.load, i32.store~~ ~~(with support for offset= parameter!)~~, ~~memory.size, memory.grow~~

- ~~i32.const, i32.add, i32.sub,~~ ~~i32.div~~, ~~i32.mul,~~ ~~i32.and, i32.or, i32.xor, i32.shl, i32.shr~~

- ~~f32.const, f32.add~~

- At least 1 float-to-int conversion

- Pick yourself from ~~trunc\_~~, convert\_, promote\_, demote\_, ~~wrap\_~~, extend\_, ~~reinterpret\_~~ functions

- ~~All i32 comparison operators (eqz, eq, ne, lt variants, gt variants, le variants, ge variants)~~

- ~~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~~ works only with local variables e.g. (local $var i32)

- ~~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~~