MUST have:

~~- 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)~~

~~- Properly handle errors/unsupported instructions~~

~~- Proper error messages + graceful exit~~

~~- Problems in 1 function/section shouldn't necessarily mean other functions can't be properly called!~~

Nice to have/expected for good score:

- Infinite loop detection

- Memory out of bounds detection

- The following instructions:

~~- i32.rem, i32.rotr, i32.rotl~~

~~- drop~~

Optional:

~~- "start" section~~

- Proper validation that the function implementations/types actually adhere to the WASM "types" section

- Other semantic validation (e.g., how many values can be left on the stack when returning from any code path)

- The following instructions:

~~- global.get, global.set, clz, ctz, popcnt~~

- i64 and f64 instruction support

- f32 support beyond what's listed above

- float-to-int conversion beyond what's listed above

- "memory" export

- memory.fill, memory.copy, memory.init

- data.drop

- any instructions not mentioned explicitly in this assignment

Not required:

- Advanced features listed in the introduction above

- Memory alignment details (i.e., the align= argument with memory operations)

- Proper interfaces with outside programming environment (e.g., actually calling C++ functions directly from inside the WASM VM)

- The following instructions:

- br\_table, call\_indirect, select, table.get, table.set,

- .load/store variants with suffixes (e.g., 8\_s, 16\_u etc.)

- ref.\*

- table.\*

- v128.\* and instructions with an "x" in the name (e.g., i8x16)