Assembler Description

The input/output format for the assembler is a text file. Each line of the text file may be of one of 3 types:

- Empty line: will these lines.
- A label.
- An instruction.
- A variable definition.

Each of these entities has the following grammar:

- The syntax of all the supported instructions is given the ISA DOCUMENT. The fields of an instruction are whitespace separated. The instruction itself might also have whitespace before it. An instruction can be one of the following:
  - The opcode must be one of the supported mnemonic.
  - A register can be one of R0, R1, ..., R6, and FLAGS.
  - A mem_addr in jump instructions must be a label.
  - An Imm must be a whole number ≤ 127 and ≥ 0.
  - A mem_addr in load and store must be a variable.
- A label marks a location in the code and must be followed by a colon (:). No spaces are allowed between the label name and colon (:).
- A variable definition is of the following format:

  `var xyz`

  which declares a 16-bit variable called `xyz`. This variable name can be used in place of mem_addr fields in load and store instructions. All variables must be defined at the very beginning of the assembly program.
- The assembler reads the assembly program as an input text file (test_file.txt).
- The assembler generates the binary output as an output text file(`assembly_output.txt`).
- The assembler generates the error notifications, including line numbers, as an output text file (stdout). In case of multiple errors, the assembler may print any one of the errors.

The assembler for the aforementioned ISA and assembly language satisfies the following requirements:

- Handling all supported instructions.
- Handling labels.

- Handling variables.
- Ensuring that any illegal instruction results in a syntax error, including:
  - Typos in instruction or register names.
  - Use of undefined variables.
  - Use of undefined labels.
  - Illegal use of the FLAGS register.
  - Illegal immediate values (more than 7 bits).
  - Misuse of labels as variables or vice versa.
  - Variables not declared at the beginning.
  - Missing `hlt` instruction.
  - `hlt` not being used as the last instruction.
  - Generating distinct readable errors for each condition.
  - Providing a "General Syntax Error" for any other illegal usage.
- Printing out all errors with corresponding line numbers.
- Generating the binary output file if the code is error-free. If the code is error free, then the corresponding binary is generated. The binary file is a text file in which each line is a 16bit binary number written using 0s and 1s in ASCII. The assembler can write less than or equal to 128 lines.

Example Assembly Program

```
var X
mov R1 $10
mov R2 $100
mul R3 R2 R1
st R3 X
hlt
```

The above program will be converted into the following machine code:

```
0001000100001010
0001001001100100
0011000011010001
0010101100000101
1101000000000000
```