BASIC ASSEMBLER

Write and Test a basic assembler to translate assembly programs (asm) that contain *no symbols* into binary code (hack)

Input Format

Input is provided through STDIN in either of the following forms. *No symbols* are used.

- Independent assembly statements
- Complete assembly code with multiple statements which perform a certain function.

Remember to provide support for handling whitespaces (line comments, in-line comments, newline spaces , indentation etc)

NOTE: Input ends with new-line character. Eg:

```
@16 // line
D=M  // line
   // new-line
```

Constraints

- Firstly, there are 3 Sample test cases (visible) so that you can *RUN* and debug your program. These are worth ***0 Points*** in total.
- Then, there are 4 Graded test cases (hidden) on which your program will be evaluated after clicking *SUBMIT*. These are worth ***35 Points*** in total.
- No Syntax Error in any of the test cases

Output Format

1. Output must be 16-bit binary code with reference to HACK assembly reference sheet.
2. Output must be written onto STDOUT

Sample Input 0

```
@1
```

Sample Output 0

```
0000000000000001
```

Explanation 0

- Input is an A-instruction.
- Output should be of format *0value* where value is a 15-bit binary constant.
- Refer to HACK reference Sheet for conversion.

Sample Input 1

BASIC ASSEMBLER

```
D=M+1
```

Sample Output 1

```
1111110111010000
```

Explanation 1

- Input is a C-instruction.
- Output should be of format *111accccccdddjjj*
- Refer to HACK reference sheet for conversion.

Sample Input 2

```
// Sample test to handle in-line comments, line comments, newline spaces, indentation
// Also has multiple assembly statements

  @20 //Instruction address is 20

  D=D+1;JGT //Assign D+1 to D. JUMP to ROM[20] if D+1>0
```

Sample Output 2

```
0000000000010100
1110011111010001
```

Explanation 2

- Input is set of assembly statements and whitespaces.
- Output should be binary code for assembly statements only.
- Refer to HACK reference sheet for conversion.