1. A partir das definições regulares abaixo que reconhecem nomes de arquivos com um caminho (opcional), escreva uma especificação JFLEX que reconhece as entradas válidas.

```
PathFileName → ( Drive : )? (\)? (PathName \ )* FileName (. FileType)?
Drive → letter
PathName → id
FileName → id
FileType → id
Id → (letter | digit )( letter | digit )*

digit: caracteres de ‘0’ a ‘9’
letter: os caracteres do alfabeto ‘a’ a ‘z’, ‘A’ a ‘Z’, “digit” ou o caracter “_”
```

Exemplos de entradas:

```
D:\acad\disc\compil\compil_u02_lexico
ExercicioLexico.doc
\Windows\System32
```

2. Escreva uma especificação JFLEX que recebe um arquivo “Java” e reconhece seus principais elementos (por exemplo: palavras reservadas, identificadore, strings, constantes numéricas, strings, comentários e imports) e gera uma saída em html onde cada um destes elementos é apresentado em uma cor diferente. Sugestão, utilize o mesmo padrão de cores da sua IDE preferida.

3. dada a especificação sintática abaixo que reconhece arquivos JSON (versão simplificada), escreva uma especificação léxica JFLEX que permita reconhecer lexixamento o programa exemplo apresentado. Para cada token reconhecido mostre o código do token e seu *lexeme,* além do número de linha onde foi reconhecido.

```
JSON → ARRAY
       | OBJECT

OBJECT →  "{" MEMBERS "}"

MEMBERS → STRING ":" VALUE
       | STRING ":" VALUE "," MEMBERS

ARRAY→ "[" ELEMENTS "]"

ELEMENTS→ ELEMENTS "," VALUE
       | VALUE

VALUE→  STRING | NUMBER | OBJECT | ARRAY
```

Teste com a entrada abaixo:

```
{
    "id": 1,
    "name": "Toner para Impressora XK 4532",
    "price": 219.23,
    "tags": [ "Toner", "4532" ],
    "stock": {
        "shopping iguatemi": 3
    }
}
```