Task 1. Para este taller vamos a construir un lexer para el lenguaje Grace. El objetivo del lexer es poder leer un programa (ingresado como una cadena de caracteres) y retornando una cadena de tokens, es decir una representación de un único char para cada una de las palabras en el lenguaje inicial.

El trabajo de este quiz-taller es definir un autómata con respuesta (en las transiciones o estados) para la tokenización de las estructuras de control de Grace (*i.e.,* ifs y loops). Adicionalmente tendremos en cuenta variables numéricas con instrucciones de operaciones básicas (*e.g.,* sumar, restar, multiplicar dividir), comparaciones booleanas, asignaciones y definiciones de variables. Para separar las instrucciones, obligaremos a que terminen con ; o }.

Es responsabilidad de ustedes definir los tokens adecuados a cada elemento. Por simplicidad podemos suponer que todas los números, variables y métodos (independientemente de su nombre) serán abstraídas por el mismo token (para cada una de ellas).

A continuación mostramos ejemplos para cada una de las estructuras de control en Grace.

Definición de ifs

```
def temperature := 4
if (temperature > 68) then {
    print 5;
} else {
    print 0;
}
```

Snippet 1: ifs

Definición de loops

```
for(1..n) do {
  i -> print i;
}
```

Snippet 2: for-loops

Por ejemplo, al utilizar el automata con respuesta como lexer de los ejemplos anteriores obtenemos las siguientes cadenas de tokens.

```
def i := 1;
def n := 10;
while {i < n} do {
  print i*i;
  i = i + 1;
}
```

Snippet 3: while-loops

$T_1VT_2NT_3(V > N)T_4T_5N;T_6T_5N;$

Snippet 4: ifs

$T_1VT_6N;T_7(NT_8V)T_9VT_{14}T_5V;$

Snippet 5: for-loops

$T_1VT_6N;T_1VT_6N;T_{10}V < VT_9T_5VT_{11}V;VT_{12}VT_{13}N;$

Snippet 6: while-loops