Algoritmos y Estructuras de Datos
Enunciado Primer Parcial Año 2024

Realizar un programa que evalúe un archivo de formato JSON-AYED. Una expresión de este tipo es una expresión delimitada por llaves que internamente tiene subexpresiones separadas por coma, donde una subexpresión es alguna de las siguientes:

- Un par (llave : valor), ambos strings delimitados por comillas dobles (”nombre”: “Carlos”)
- Un par (”llave”: lista) de strings separados por coma y delimitados por corchetes ([”string1”,..,”stringX”])
- Un par (llave : subexpresion) recursivamente definida delimitada por llaves ({})

Un ejemplo de esto es:

```
{
"nombre":‘‘Carlos",
"apellido":‘‘Garciarena",
"domicilio": {
                ‘‘calle":"San martin",
                ‘‘altura":‘‘33",
                ‘‘cuidad":‘‘Jujuy",
                "telefonos":["351-4555666","351-5012789"]
            },
"conyuge":"Ana Castro",
"hijos":["Ana Garciarena","Lucas Garciarena"]
}
```

Ud. deberá desarrollar un programa que lea un archivo de texto que tenga una sola expresión JSON-AYED, validar que sea correctamente escrita, e imprimirla en otro archivo eliminando los espacios entre expresiones, saltos

de linea y tab que tenga el archivo de entrada, o bien, si no es correcta la expresión de entrada, indicar donde se produjo el error en la expresión de entrada.

Para el desarrollo debe basarse en los siguientes lineamientos:

- Debe leer los caracteres del archivo de entrada uno a uno.
- Debe utilizar estructuras de pilas y colas con ciertas especificaciones.
- El archivo de salida no tiene espacios, tabs ni saltos de linea.

Debe utilizar una clase para cada tipo de componente de una expresión JSON-AYED:

1. Expresión entre llaves (expresión JSON-AYED)
2. Expresión llave : valor
3. Expresión llave : expresión JSON-AYED
4. Expresión llave : expresión lista de strings

Cada uno de estos componentes debe tener una pila que se utilice en la evaluación de la correctitud de la expresión. Adicionalmente la lista de strings debe tener una cola donde se almacenen los valores que tiene dicha expresión.

Para evaluar la expresión de entrada, al ir leyendo uno a uno los caracteres del archivo se entran en ciertos "modos"de avance del proceso que definen la validez del próximo caracter a leer, por ejemplo, el validador de la expresión JSON-AYED tiene una pila que esta vacía al inicio de la lectura, por lo cuanto sabe que el próximo carácter debe ser una llave que abre, y cuando lee una llave que cierra determina que la expresión terminó y tiene la pila vacía. Otro ejemplo es para leer un string, que comienza con una comilla doble, siguen caracteres alfanuméricos y espacios y debe terminar con otra comilla doble.

Se aconseja que cada "modo"tenga su propia pila para evaluación. Al cambiar de modo se activa su tipo de pila de evaluación y al volver al modo anterior se restablece el de llamada.

Cada clase a su vez debe tener un buffer de almacenamiento donde guardar el contenido que tiene cada instancia de la respectiva clase y que se utiliza luego al hacer la impresión de la expresión leída por medio de un método print que llame a los print de los objetos almacenados en el buffer.

Deberá codificar la solución usando Clases de C++, con métodos que respeten el comportamiento definido en esta especificación. Lo que no está

especificado aquí es de libre elección. Podrá utilizar el código dado en clases prácticas ajustando el tipo de dato contenido en cada estructura al tipo respectivo a la clase a la cual pertenece.