Universidad del Valle de Guatemala
Facultad de Ingeniería
Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Laboratorio No. 2

Para los siguientes incisos, siga estas instrucciones:

- Cree un repositorio **privado** en GitHub.com para alojar el código correspondiente para este laboratorio.
- Añada a los siguientes usuarios con rol de *collaborator*:
  - Aristondo01
  - Daniel14gonc
  - gbrolo
- Coloque todas las instrucciones que considere pertinentes en un archivo denominado README.md en la raíz del repositorio.
- Para los incisos que no involucren la generación de código, cree una carpeta por separado en su repositorio y coloque las respuestas en un documento con formato PDF
- Grabe un video de no más de 10 minutos donde muestre la ejecución de sus programas para los ejercicios 2 y 3. Súbalo a YouTube como video no listado y adjúntelo en el README de su repositorio.

Ejercicio No. 1 (25%) – Convierta las siguientes expresiones regulares en autómatas finitos deterministas (para ello deberá primero convertir las expresiones regulares a AFN y luego convertir a AFD). Muestre todo su procedimiento, i.e., AFN construido con Thompson, tabla de transición, conversión a AFD. Para el inciso g, interprete \ como un escape de carácter, i.e., \(significa que su regex reconoce el caracter (.

a) $(a|t)c$
b) $(a|b)*$
c) $(a*|b*)*$
d) $((\varepsilon|a)|b*)*$
e) $(a|b)*abb(a|b)*$
f) $0?(1?)?0*$
g) $if\backslash([ae]+\backslash)\backslash\{[ei]+\backslash\}(\backslash n(else\backslash\{[jl]+\backslash\}))?$
h) $[ae03]+@[ae03]+.(com|net|org)(.(gt|cr|co))?$

Ejercicio No. 2 (25%) – Escriba código en el lenguaje de programación de su gusto para implementar un algoritmo capaz de balancear expresiones en formato infix. Implemente el uso de una pila para llevar track de los símbolos de interés, i.e., (), [], {}, para definir el buen balanceo.

La ejecución de su programa debe ser la siguiente:

- Su programa debe leer un archivo de texto y procesar cada línea en este archivo.
- Por línea procesada, su programa deberá de indicar si la expresión regular en cuestión se encuentra bien balanceada o no.
- Deberá mostrar la secuencia de pasos de la pila para validar que el algoritmo implementado use esta misma de forma adecuada para validar el buen balanceo de la expresión.
- En el video que adjunte para su ejecución, asegúrese de mostrar en pantalla el archivo a ejecutar y el resultado de la ejecución y cada uno de sus pasos y que se refleje en la grabación que ese archivo está siendo ejecutado y no es otro o es código *hardcoded*.
- Utilice las siguientes expresiones regulares (una por línea en el archivo):
  - $a(a|b)*b+a?$

- $\big((a|b)b\big) * [az]b]$
- $(a * b * c * d * (a|e|i|o|u))e * f * g * h)\{1,2\}$
- $\text{^}[aZ].com\{5,30\}$
- $([[az][AZ]](((((.|;)|;)|.)|.)|.)\{10,20\}) *) +$

- De no cumplir con los rubros anteriores la nota de este inciso será de 0 puntos.

Ejercicio No. 3 (50%) – Escriba código en el lenguaje de programación de su gusto para implementar el algoritmo de Shunting Yard para convertir expresiones regulares en notación infix a notación postfix. Siga las siguientes instrucciones:

- Investigue sobre el algoritmo de Shunting Yard y provea una breve explicación escrita de cómo funciona el algoritmo.
- Implemente en código de alto nivel el algoritmo en cuestión, para lo que deberá recibir como input un archivo de texto con expresiones regulares por línea. Procese cada línea en el archivo, dando como output la siguiente información:
  - La expresión en formato postfix.
  - Los pasos realizados para llegar a la conversión a postfix.
- Utilice como referencia el siguiente pseudocódigo: https://gist.github.com/gbrolo/1a80f67f8d0a20d42828fb3fdb7be4de
  - Se recomienda altamente que utilice este pseudocódigo por el factor tiempo, pero se incita a que comprenda en realidad cómo funciona el algoritmo.
- Las expresiones regulares que deberá de utilizar para este inciso son las mismas del Ejercicio No. 1.
  - Deberá implementar en su algoritmo un verificador de caracteres escapados por \.
  - Tenga cuidado con el pseudo código, este es una versión muy sencilla de Shunting Yard y utiliza caracteres comunes para expresar concatenaciones explícitas, por ejemplo, como el uso de “.”
  - Esto le dará problemas al procesar expresiones regulares con “.”, por ejemplo.
  - Deberá implementar también una validación para convertir las extensiones de expresiones regulares tales como “+” y “?” para el correcto funcionamiento del algoritmo.
- De no cumplir con los rubros anteriores la nota de este inciso será de 0 puntos.