



**Vacaciones Junio 2023**

Sección	Catedrático	Tutor Académico
P	Ing. Kevin Adiel Lajpop Ajpacaja	Elder Anibal Pum Rojas

**Enunciado Proyecto 2**  
**SPARK STACK**

**Objetivos:**

- Implementar un software en Python que permitan plasmar los conocimientos sobre lenguajes independientes del contexto, adquiridos en la unidad 3 del programa del curso de lenguajes formales y de programación.
- Dar continuidad a la implementación de soluciones de software empleando paradigmas de programación.
- Desarrollar habilidades de comprensión sobre gramáticas y autómatas

**Descripción:**

Para poder profundizar más en el tema de los **lenguajes independientes del contexto** se solicita que realice un programa con **Interfaz Gráfica** utilizando el lenguaje de programación **Python**. El programa llamado "**Spark Stack**" integra dos apartados de funcionalidades, uno enfocado en **gramáticas libres del contexto** y otro más que integra funcionalidades útiles para la comprensión de los **autómatas de pila**.

El flujo del programa funcionará a través de menús con los cuales el usuario podrá navegar entre las distintas funcionalidades que ofrecen los módulos. Cada uno de los dos módulos agrupan acciones de gran utilidad para quienes desean comprender el funcionamiento de los lenguajes libres del contexto.

La creación de los autómatas de pila, como de gramáticas libres del contexto, será realizada a través de archivos de entrada que permitirán optimizar el funcionamiento del programa. Cabe resaltar que los archivos de entrada tendrán una estructura definida y sin errores, de manera que puedan ser leídos utilizando funcionalidades propias de Python sin necesidad de implementar algún tipo de analizador.

Durante la calificación se harán preguntas acerca de la manera en la cual se implementó cada una de las funcionalidades para poder comprobar que el estudiante sea el autor de la solución.

## Características de “Spark Stack”

### Pantalla inicial

Cuando se inicie el programa se deberá mostrar una pantalla de bienvenida que muestre el nombre del programa y los datos esenciales del desarrollador (Nombre, Carnet), así también en la parte inferior deberá mostrar una cuenta regresiva de 5 segundos, cuando la cuenta finalice, la pantalla de inicio deberá eliminarse automáticamente y mostrar el menú principal.

### Menú principal

El menú principal es la pantalla que permite seleccionar el módulo con el cual se desea trabajar. Para seleccionar una opción, el usuario deberá de seleccionar por medio de un conjunto de botones, la opción que él requiera.

### Opciones del menú principal

1. **Modulo gramática libre de contexto:** Al seleccionar esta opción se mostrarán las opciones disponibles para trabajar con gramáticas libres de contexto.
2. **Módulo autómatas de pila:** Esta opción conducirá al usuario al módulo que contiene las opciones para trabajar con autómatas de pila.
3. **Salir:** Al seleccionar esta opción se debe mostrar una pantalla de despedida con una cuenta regresiva de 5 segundos, al finalizar los 5 segundos, el programa se debe cerrar.

## Descripción de los módulos

### MODULO DE GRAMÁTICAS LIBRES DE CONTEXTO

Este módulo permite la gestión de gramáticas libres de contexto, las opciones que presentará son las siguientes:

#### **1. Cargar archivo**

Esta opción permite cargar N gramáticas libres del contexto en el sistema. Spark Stack debe tener la capacidad de verificar que la gramática leída sea una gramática libre de contexto, en caso de que la gramática no cumpla con las características específicas de una gramática libre de contexto deberá descartarla, no la cargará en el sistema y continuará con la siguiente.

La estructura del archivo de entrada será la siguiente: los componentes de la gramática estarán especificados línea a línea y el final de la gramática estará marcado con un símbolo %.

A continuación, se muestra un ejemplo:

Ubicación	Valor	Ejemplo de archivo
Línea 1	Nombre	Grm1
Línea 2	No Terminales	S,A,B,C
Línea 3	Terminales	a,b,z
Línea 4	No terminal inicial	S
Línea 5	Producción 1	S::=A
Línea 6	Producción 2	A::=a A a
Línea 7	Producción 3	A::=B
Línea 8	Producción 4	B::=b B b
Línea 9	Producción 5	B::=C
Línea 10	...	C::=z C
Línea 11	Producción n	C::=z
Línea 12	Fin	%
Línea 13	Nombre	Gramatica2
Línea 14	No terminales	A,B,C,D
Línea 15	Terminales	0,1
Línea 16	No terminal inicial	A
Línea 17	Producción 1	A::=1 B
Línea 18	Producción 2	A::=0 C
Línea 19	Producción 3	B::=1 A
Línea 20	Producción 4	B::=0 D
Línea 21	Producción 5	C::=1 D
Línea 22	Producción 6	C::=0
Línea 23	...	D::=1 C
Línea 24		C::=0 A
Línea 25	Producción n	D::=0 B
Línea 26	Fin	%
Línea ...		
Línea...		

#### Consideraciones:

Para el ingreso de producciones desde archivo, se suprimió el uso del símbolo “|” para una mayor legibilidad. Si un “no terminal” tiene más de una derivación aparecerá la regla completa con el símbolo “>”. Lo que implica que las producciones pueden venir en desorden.

Si la gramática no posee ninguna producción que la identifique únicamente como gramática independiente del contexto deberá descartarse. En el ejemplo anterior la segunda gramática debería descartarse porque solamente reúne las características de una gramática regular.

Cuando finaliza la carga del archivo, se debe retornar al menú del módulo gramáticas.

El nombre especificado para la gramática servirá para utilizar la gramática en las demás opciones del módulo.

La extensión de los archivos de entrada será .glc

## 2. Mostrar información general

Esta opción deberá mostrar todos los nombres de gramáticas que se encuentran actualmente en el sistema para que el usuario pueda elegir una. Cuando el usuario elija una gramática, inmediatamente se deberá mostrar la información de la gramática en la interfaz gráfica. Los datos que deben mostrarse son los siguientes:

Nombre, No terminales, Terminales, No terminal inicial, producciones.

Las producciones deben ser mostradas con la siguiente notación:

```
No terminal > Expresión
           | Expresión
No terminal > Expresión
No terminal > Expresión
```

### Consideraciones:

Se debe utilizar el símbolo “|” para identificar cuando un no terminal tiene más de una posible derivación.

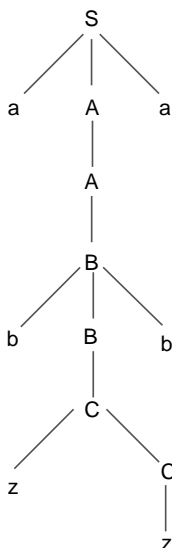
La gramática solo debe mostrarse en la interfaz, no es necesario generar un reporte externo. Después de mostrar la gramática se debe dejar en espera el programa y cuando el usuario lo indique, el programa debe limpiar la pantalla y mostrar el menú del módulo de gramáticas.

“Expresión” representa las combinaciones de terminales y no terminales permitidas en las gramáticas libres de contexto.

## 3. Árbol de derivación

Se debe solicitar al usuario el nombre de la gramática que desea graficar. Cuando el usuario proporcione el nombre, se deberá generar el árbol de derivación respectivo para la gramática.

Ejemplo de salida para la gramática “Grm1” del ejemplo de archivo de entrada:



### Consideraciones:

El grafo debe visualizarse automáticamente.

Para una mayor rapidez este reporte debe ser generado en un archivo de imagen.

El grafo debe generarse de tal forma que pueda ser comprendido.

## MÓDULO DE AUTÓMATAS DE PILA

Como se mencionaba en la sección anterior, Spark Stack es un software que permite agilizar el trabajo con autómatas de pila. Por esta razón posee un módulo que integra funcionalidades específicas para este tipo de autómatas. A continuación, se muestran las opciones con las que debe contar el módulo:

### 1. Cargar archivo

Esta opción permite cargar N autómatas de pila en el sistema. Los componentes del autómata estarán especificados línea a línea y el final de las transiciones del autómata estará marcado con un símbolo %. El orden de las líneas que contienen los respectivos datos se describe a continuación:

Ubicación	Valor	Ejemplo de archivo
Línea 1	Nombre	AP1
Línea 2	Alfabeto	a,b
Línea 3	Símbolos de pila	a,b,#
Línea 4	Estados	I,A,B,C,F
Línea 5	Estado inicial	I
Línea 6	Estado de aceptación	F
Línea 7	Transición 1	I,\$,\$;A,#
Línea 8	Transición 2	A,a,\$;B,a
Línea 9	Transición 3	B,a,\$;B,a
Línea 10	Transición 4	B,b,a;C,\$
Línea 11	Transición 5	C,b,a;C,\$
Línea 12	Transición 6	C,\$,#;F,\$
Línea 13	Fin	%
Línea 14	Nombre	AP2
Línea 15	Alfabeto	x,y
Línea 16	Símbolos de pila	x,y,#
Línea 17	Estados	S0,S1,S2,S3
Línea 18	Estado inicial	S0
Línea 19	Estado de aceptación	S3
Línea 20	Transición 1	S0,\$,\$;S1,#
Línea 21	Transición 2	S1,x,\$;S1,x
Línea 22	Transición 3	S1,y,x;S2,\$
Línea 23	Transición 4	S2,y,x;S2,\$
Línea 24	Transición 5	S2,\$,#;S3,\$
Línea 25	Fin	%
Línea 26	Nombre ...	...
Línea ...		
Línea...		

#### Consideraciones:

Pueden venir N autómatas de pila en el archivo.

El símbolo % representa el final de las transiciones de un autómata de pila.

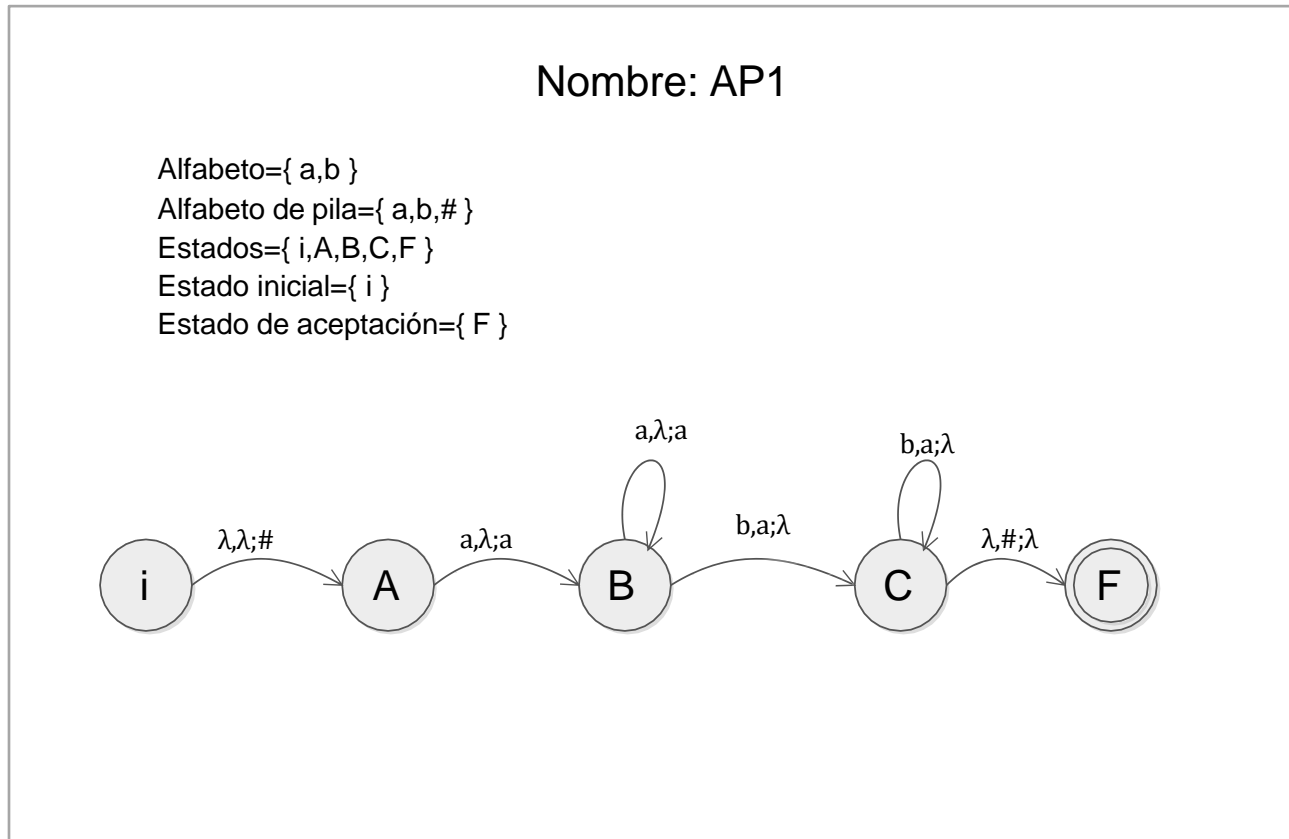
Se sustituyó el símbolo  $\lambda$  por \$ para evitar posibles problemas de legibilidad en la aplicación. Las transiciones tienen el siguiente formato:

ESTADO ORIGEN, SÍMBOLO QUE LEE DE LA ENTRADA, SÍMBOLO QUE EXTRAER DE LA PILA, ESTADO DESTINO, SÍMBOLO QUE INSERTA EN PILA

## 2. Mostrar información de autómeta

Esta opción servirá para verificar que el autómeta fue almacenado correctamente, le solicitará al usuario el nombre de un autómeta a mostrar, luego de que el usuario ingrese el nombre correcto se deberá mostrar un reporte en PDF con todos los datos del autómeta.

A continuación, se muestra el reporte que se esperaría para el autómeta "AP1" del archivo de entrada ejemplo:



### Consideraciones:

El reporte deberá generarse en pdf.

Se debe identificar cada una de las transiciones del autómeta de pila.

El reporte debe incluir la información de cada uno de los componentes del autómeta.

En caso de que no se pueda mostrar el carácter  $\lambda$  correctamente puede utilizarse \$ en su defecto.

Las transiciones tienen el siguiente formato:

SÍMBOLO QUE LEE DE LA ENTRADA, SÍMBOLO QUE EXTRAE DE LA PILA, SÍMBOLO QUE INSERTA EN PILA

### 3. Validar una cadena

Esta opción solicitará al usuario el nombre de un autómata de pila que deberá estar previamente cargado en el sistema. Finalizada la elección del autómata, se solicitará al usuario el ingreso de una cadena entrada para que sea validada con el autómata. El autómata deberá leer la cadena y solamente mostrará en la interfaz si pudo validar o no pudo validar la entrada.

### 4. Ruta de validación

Esta opción solicitará al usuario el nombre de un autómata de pila que deberá estar previamente cargado en el sistema. Finalizada la elección del autómata, se solicitará al usuario el ingreso de una cadena entrada para que sea validada con el autómata. El autómata deberá leer la cadena y si la entrada es reconocida deberá mostrar la lista de transiciones que siguió para determinar su validez.

Ejemplo de salida al seleccionar el autómata “AP1” e ingresar la entrada aabb.

```
La cadena fue reconocida exitosamente:
```

```
Ruta:
```

```
i,$,$;#,A
A,a,$;a,B
B,a,$;a,B
B,b,a;#,C
C,b,a;#,C
C,$,#;$,F
```

#### Consideraciones:

Si la cadena no es válida solamente deberá mostrar que no fue posible validar la entrada.

El símbolo \$, sustituye a  $\lambda$  en las salidas en interfaz para evitar problemas de legibilidad.

Las transiciones deben tener el formato definido en las consideraciones de la carga de archivo.

### 5. Recorrido paso a paso

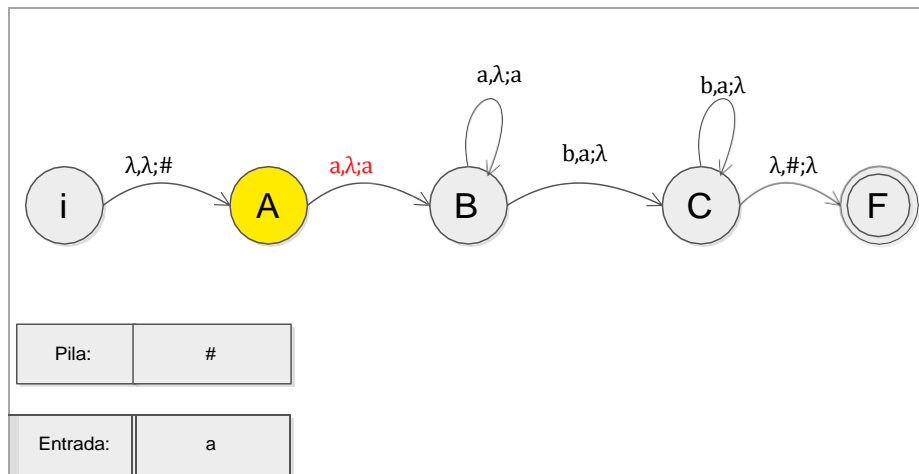
Esta opción solicitará al usuario el nombre de un autómata de pila que deberá estar previamente cargado en el sistema. Finalizada la elección del autómata, se solicitará al usuario el ingreso de una cadena entrada para que sea validada con el autómata. Esta opción permitirá visualizar el comportamiento del autómata un paso a la vez, mostrando un archivo de imagen que ilustre los pasos como se mostrará a continuación:

**Ejemplo de la función paso a paso utilizando el autómata “AP1” y la cadena de entrada “aabb”:**

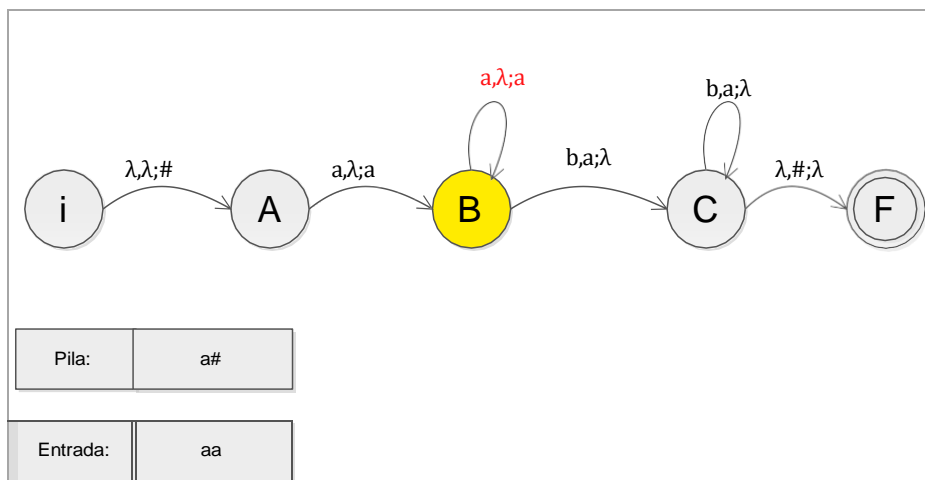
Luego de ingresar los datos que se mencionan, Spark Stack debería mostrar como primer paso el estado inicial de la pila, la entrada, y marcar el estado actual del autómata:



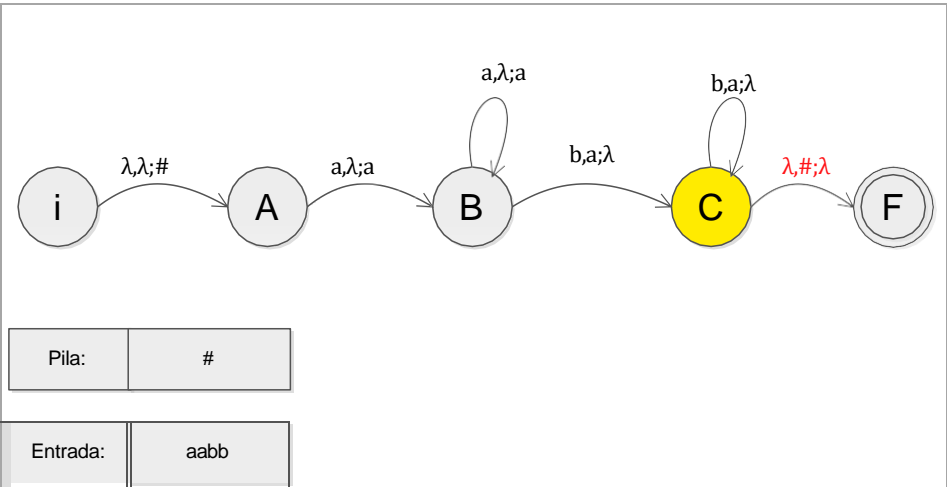
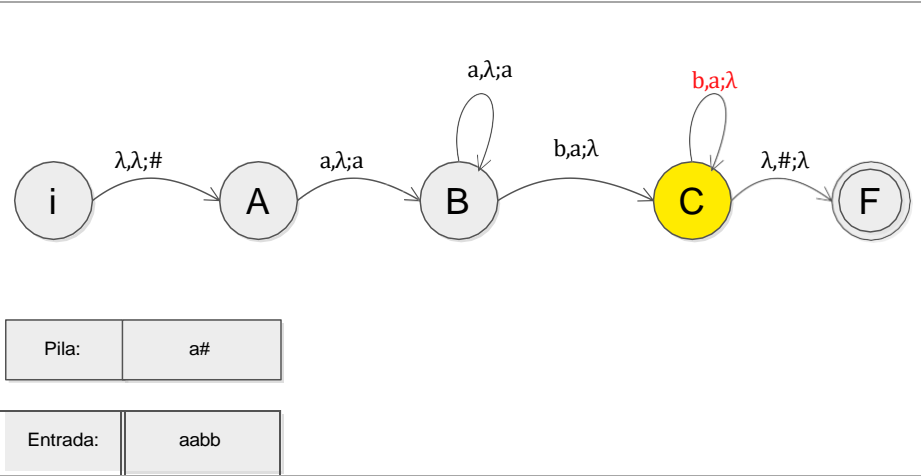
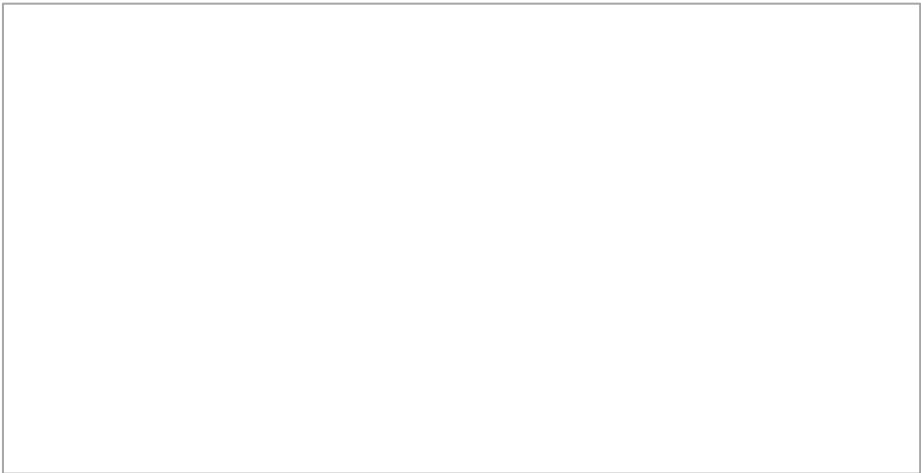
Cuando el usuario cierre la imagen, el programa debe mostrar inmediatamente el siguiente paso:



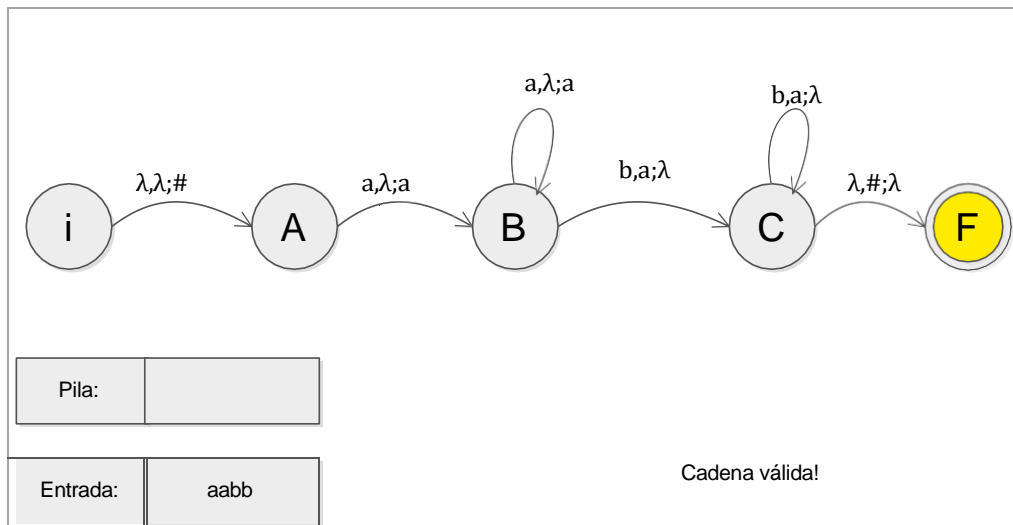
Las siguientes iteraciones de este ejemplo se verían de la siguiente manera:







Hasta que finalmente se muestre el resultado:



### Consideraciones:

Para un mejor desempeño del programa cada iteración debe mostrar el resultado de la iteración en un **archivo de imagen** que se abra automáticamente.

Queda a discreción del estudiante la manera en que mostrará la pila y las transiciones, la única condición es que debe mostrarse de forma clara lo que está ocurriendo en ese momento con el autómata y sus componentes.

En caso de que no se pueda mostrar el carácter  $\lambda$  correctamente puede utilizarse  $\$$  en su defecto.

## 6. Validar cadena en una pasada

Esta opción solicitará al usuario el nombre de un autómata de pila que deberá estar previamente cargado en el sistema. Finalizada la elección del autómata, se solicitará al usuario el ingreso de una cadena entrada para que sea validada con el autómata. Esta opción deberá mostrar como resultado un reporte en pdf con una tabla de resumen.

Ejemplo de salida esperada utilizando el autómata AP1 y la entrada aabb:

Iteración	Pila	Entrada	Transición
0			$i, \lambda, \lambda; \#, A$
1	#	a	$A, a, \lambda; a, B$
2	a#	aa	$B, a, \lambda; a, B$
3	aa#	aab	$B, b, a; \lambda, C$
4	a#	aabb	$C, b, a; \lambda, C$
5	#	aabb	$C, \lambda, \#; \lambda, F$
6		Aabb	

### Consideraciones:

El reporte debe generarse en pdf.

En caso de que no se pueda mostrar el carácter  $\lambda$  correctamente puede utilizarse  $\$$  en su defecto. Los datos de la tabla deben mostrarse de forma legible.

## MOCKUPS DE REFERENCIA

### 1. Pantalla Inicial

Proyecto 2
<p>Nombre del Curso: Lab. Lenguajes Formales y de Programación B+</p> <p>Nombre del Estudiante: xxxxxxxxx</p> <p>Carné del Estudiante: xxxxxxxx</p> <div>Esta pantalla se destruirá en 5 segundos.</div>

### 2. Menú Principal

Menú Principal
<div>Módulo Gramáticas</div> <div>Módulo Autómatas</div> <div>Salir</div>

### 3. Módulo Gramáticas

Módulo Gramáticas
<div>Cargar Archivo</div> <div>Información General</div> <div>Árbol de Derivación</div> <div>Atrás</div>

#### 4. Módulo Autómatas

Módulo Autómatas

Cargar Archivo

Ruta de Validación

Información General

Paso a Paso

Validar Cadena

Una Pasada

Atrás

#### 5. Cargar Archivo

Cargar Archivo

Ruta:

Seleccionar

Atrás

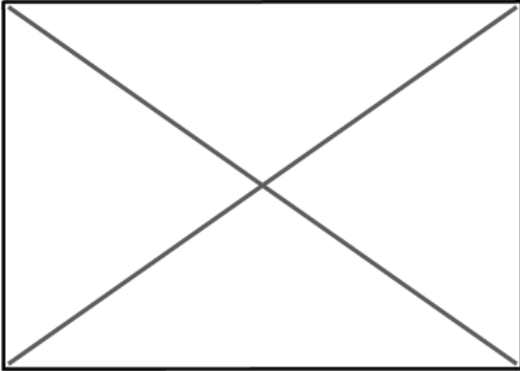
#### 6. Información General

Información General Autómata

Atrás

## 7. Árbol de Derivación

Árbol de Derivación



Atrás

## 8. Validar Cadena

Validar Cadena

Autómata

Cadena

Validar

Atrás

## 9. Paso a Paso

Paso a Paso

Autómata

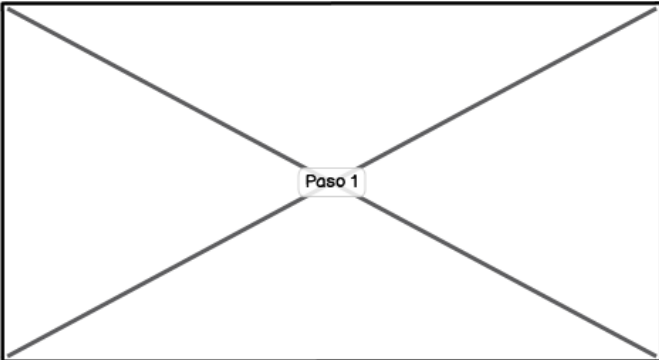
Cadena

Validar

Siguiente

Atrás

Anterior



Paso 1

## 10. Una pasada

Una Pasada

Autómata

Seleccionar

Validar

Cadena

Atrás

Iteración	Pila	Entrada	Transición
0			$i \lambda \lambda_i \# A$
x	xxxxxx	xxxxxx	xxxxxx
x	xxxxxx	xxxxxx	xxxxxx
x	xxxxxx	xxxxxx	xxxxxx
x	xxxxxx	xxxxxx	xxxxxx
x	xxxxxx	xxxxxx	xxxxxx
x	xxxxxx	xxxxxx	xxxxxx

### Entregables:

- Manual de Usuario (Markdown)
- Manual Técnico (Markdown)
- Código Fuente
- Repositorio de Github

### Notas importantes:

- El proyecto se debe desarrollar de forma individual.
- Este proyecto se deberá desarrollar utilizando Python y Graphviz.
- La entrega se realizará en la plataforma UEDI.
- Se deberá de crear un repositorio de Github con el siguiente formato: **LFP\_P2\_carnet**, el estudiante es responsable de verificar que dentro del repositorio se encuentren todos los archivos necesarios para su calificación.
- No será permitida la modificación de archivos de entrada durante la calificación.
- La calificación deberá ser en línea y se estará grabando la reunión para tener un respaldo de la forma en que se procedió.
- La calificación del proyecto será personal y durará como máximo 30 minutos, en un horario que posteriormente será establecido.
- No se dará prórroga para la entrega del proyecto.
- **Copia parcial o total del proyecto tendrá una nota de 0 puntos, y se notificará a la escuela de sistemas para que se apliquen las sanciones correspondientes.**
- **En el caso de no cumplir con alguna de las indicaciones antes mencionadas, NO se calificará el proyecto; por lo cual, se tendrá una nota de cero puntos.**

**Fecha de entrega: 30 de junio de 2023 antes de las 23:59.**