

Instituto Politécnico Nacional

Escuela Superior de Cómputo

Teoría Computacional

Reporte segundo bloque de programas

Alumno: Javier Said Naranjo Miranda

Grupo: 2CM4

Índice

1. Automata Web-Ebay	2
1.1. Descripción del programa	2
1.2. Código	2
1.3. Pruebas	8
2. Balanceo de Paréntesis	11
2.1. Descripción del programa	11
2.2. Código	11
2.3. Pruebas	15
3. Palindromos-Gramática libre de contexto	17
3.1. Descripción del programa	17
3.2. Código	17
3.3. Pruebas	21

1. Automata Web-Ebay

1.1. Descripción del programa

El siguiente programa reconoce todas las palabras que contengan web o ebay, esto se realiza por medio de un automata finito determinista, podrá reconocer las palabras solo en textos en inglés.

Consta de un modo automático y manual, además se podrá visualizar el diagrama del autómata.

El programa guarda todas las palabras que encontró y las guarda en un archivo de texto indicando el número de fila y palabra en la que se encuentra.

1.2. Código

El código utilizado para la resolución del problema se muestra a continuación:

Código: webay.py

```
def automataWebay(caracter, estado, archivo):
    if(estado==0):
        estado=estadoCero(caracter, archivo)
    elif(estado==1):
        estado=estadoUno(caracter, archivo)
    elif(estado==2):
        estado=estadoDos(caracter, archivo)
    elif(estado==3):
        estado=estadoTres(caracter, archivo)
    elif(estado==4):
        estado=estadoCuatro(caracter, archivo)
    elif(estado==5):
        estado=estadoCinco(caracter, archivo)
    elif(estado==6):
        estado=estadoSeis(caracter, archivo)
    elif(estado==7):
        estado=estadoSiete(caracter, archivo)

    return estado

def estadoCero(caracter, archivo):
    if(caracter=='w'):
        archivo.write('q0—w—>q1\t')
        return 1
    elif(caracter=='e'):
        archivo.write('q0—e—>q4\t')
        return 4
    else:
        archivo.write('q0— %s—>q0\t' %caracter)
        return 0
```

```

def estadoUno(caracter, archivo):
    if(caracter=='w'):
        archivo.write('q1—w—>q1\t')
        return 1
    elif(caracter=='e'):
        archivo.write('q1—e—>q2\t')
        return 2
    else:
        archivo.write('q1— %s—>q0\t' %caracter)
        return 0

def estadoDos(caracter, archivo):
    if(caracter=='w'):
        archivo.write('q2—w—>q1\t')
        return 1
    elif(caracter=='e'):
        archivo.write('q2—e—>q4\t')
        return 4
    elif(caracter=='b'):
        archivo.write('q2—b—>q3\t')
        return 3
    else:
        archivo.write('q2— %s—>q0\t' %caracter)
        return 0

def estadoTres(caracter, archivo):
    if(caracter=='w'):
        archivo.write('q3—w—>q1\t')
        return 1
    elif(caracter=='e'):
        archivo.write('q3—e—>q4\t')
        return 4
    elif(caracter=='a'):
        archivo.write('q3—a—>q6\t')
        return 6
    else:
        archivo.write('q3— %s—>q0\t' %caracter)
        return 0

def estadoCuatro(caracter, archivo):
    if(caracter=='w'):
        archivo.write('q4—2—>q1\t')
        return 1
    elif(caracter=='e'):
        archivo.write('q4—e—>q4\t')
        return 4
    elif(caracter=='b'):
        archivo.write('q4—b—>q5\t')
        return 5
    else:
        archivo.write('q4— %s—>q0\t' %caracter)

```



```

        Leer_texto\n3.–Diagrama\n4.–Salir\nElija_una_opcion:_')
opcion=int(opcion)
except:
    print('\nIntroduzca_una_opcion_correcta\n')

return opcion

def IniciarArchivo():
    archivo=open("Palabras.txt","w")
    archivo.close
    archivoH=open("Historia.txt","w")
    archivo.close

def AbrirArchivo():
    try:
        archivo=open("Palabras.txt","a")
    except:
        print("\nError_al_abrir_el_archivo")
        exit()
    return archivo

def AbrirHistoria():
    try:
        archivo=open("Historia.txt","a")
    except:
        print("\nError_al_abrir_el_archivo")
        exit()
    return archivo

def escribir(archivo,palabra_aux,no_palabra,no_fila):
    archivo.write("Palabra:_"+palabra_aux+"_Numero_de_palabra:_"+str(
        no_palabra)+"_Numero_de_fila:_"+str(no_fila))
    archivo.write("\n")

def Evaluar(texto):
    archivo=AbrirArchivo()
    historia=AbrirHistoria()
    palabra_aux=''
    final=False
    estado=0
    no_palabra=1
    no_fila=1

    historia.write("\n\n\n\n")
    for caracter in texto:
        caracter=caracter.lower()
        estado=webay.automataWebay(caracter,estado,historia)
        if(estado==3 or estado==7):
            final=True
        if(caracter=='\n'):
            if(final):

```

```

        escribir (archivo , palabra_aux , no_palabra , no_fila)
        historia . write ( "\n" )
        palabra_aux = ''
        final = False
    else :
        palabra_aux = ''
        no_palabra = 1
        no_fila = no_fila + 1
    continue
if (caracter == '_') :
    if (final) :
        escribir (archivo , palabra_aux , no_palabra , no_fila)
        historia . write ( "\n" )
        palabra_aux = ''
        final = False
    else :
        palabra_aux = ''
        no_palabra = no_palabra + 1

    continue
palabra_aux = palabra_aux + caracter
if (final) :
    no_palabra = no_palabra + 1
    escribir (archivo , palabra_aux , no_palabra , no_fila)
    historia . write ( "\n" )

archivo . close
historia . close
def leer_Archivo () :
    try :
        archivo = open ( " archivo . txt " , " r " )
        texto = str ( archivo . read () )
    except :
        print ( "\nError_al_abrir_el_archivo" )
        exit ()
    return texto

def main () :
    IniciarArchivo ()
    while True :
        eleccion = menu ()
        if (eleccion == 1) :
            texto = input ( " Introduzca_un_pequenio_texto : _ " )

            Evaluar ( texto )
            print ( " Evaluacion_terminada , _cheque_el_archivo_de_texto " )
            while True :
                reop = input ( " Desea_regresar_al_menu\n1. - Si\n2. - No\nEleccion : _ " )
                if (reop == '1') :
                    break
                elif (reop == '2') :
                    exit ()

```

```

        else:
            continue
    elif (eleccion==2):
        texto=leer_Archivo()

        Evaluar(texto)
        print("Evaluacion_terminada,_cheque_el_archivo_de_texto")
        while True:
            reop=input("Desea_regresar_al_menu\n1.-Si\n2.-No\nEleccion:_")
            if (reop=='1'):
                break
            elif (reop=='2'):
                exit()
            else:
                continue
    elif (eleccion==3):
        diagrama.mostrarDiagrama()
        while True:
            reop=input("Desea_regresar_al_menu\n1.-Si\n2.-No\nEleccion:_")
            if (reop=='1'):
                break
            elif (reop=='2'):
                exit()
            else:
                continue
    elif (eleccion==4):
        exit()
    else:
        continue
main()

```

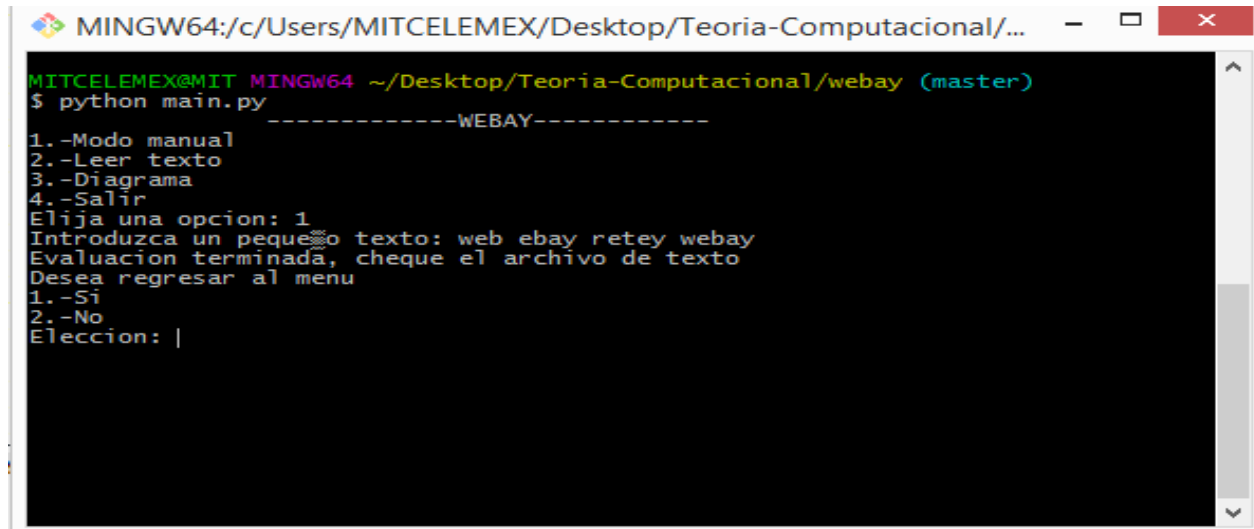
Código:diagrama.py

sjdkasjdkasjksa

1.3. Pruebas

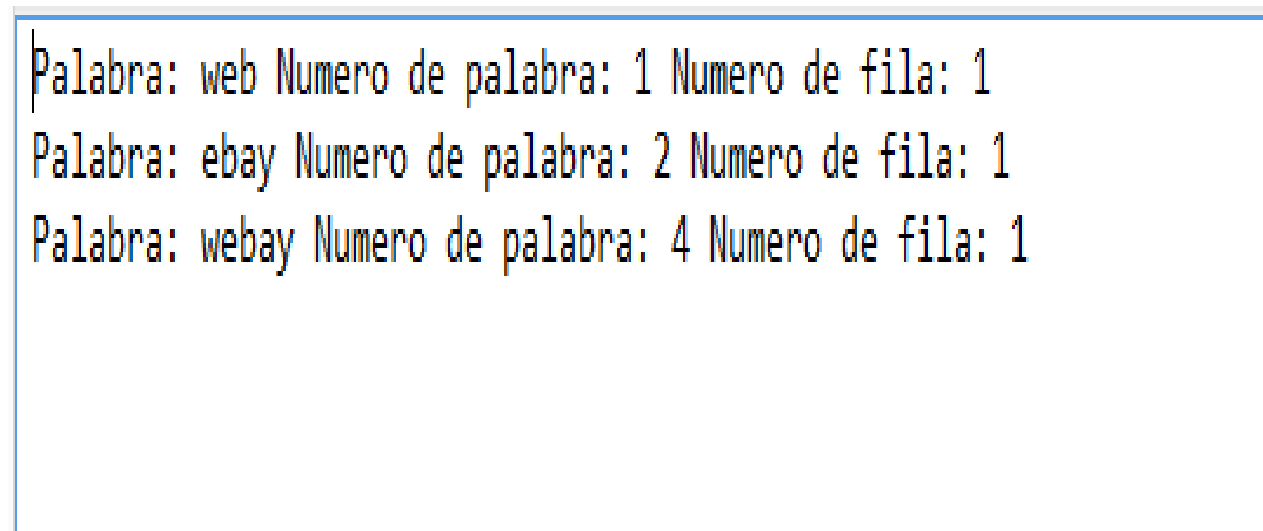
A continuación se mostraran algunas imágenes capturadas al momento de ejecutar el programa, dichas imágenes mostraran los resultados obtenidos.

Para el modo manual:



```
MITCELEMEX@MIT MINGW64 ~/Desktop/Teoria-Computacional/webay (master)
$ python main.py
-----WEBAY-----
1.-Modo manual
2.-Leer texto
3.-Diagrama
4.-Salir
Elija una opcion: 1
Introduzca un pequeño texto: web ebay retey webay
Evaluacion terminada, cheque el archivo de texto
Desea regresar al menu
1.-Si
2.-No
Eleccion: |
```

Figura 1: Palabras de prueba: web ebay retey webay



```
Palabra: web Numero de palabra: 1 Numero de fila: 1
Palabra: ebay Numero de palabra: 2 Numero de fila: 1
Palabra: webay Numero de palabra: 4 Numero de fila: 1
```

Figura 2: Salida del archivo de palabras encontradas

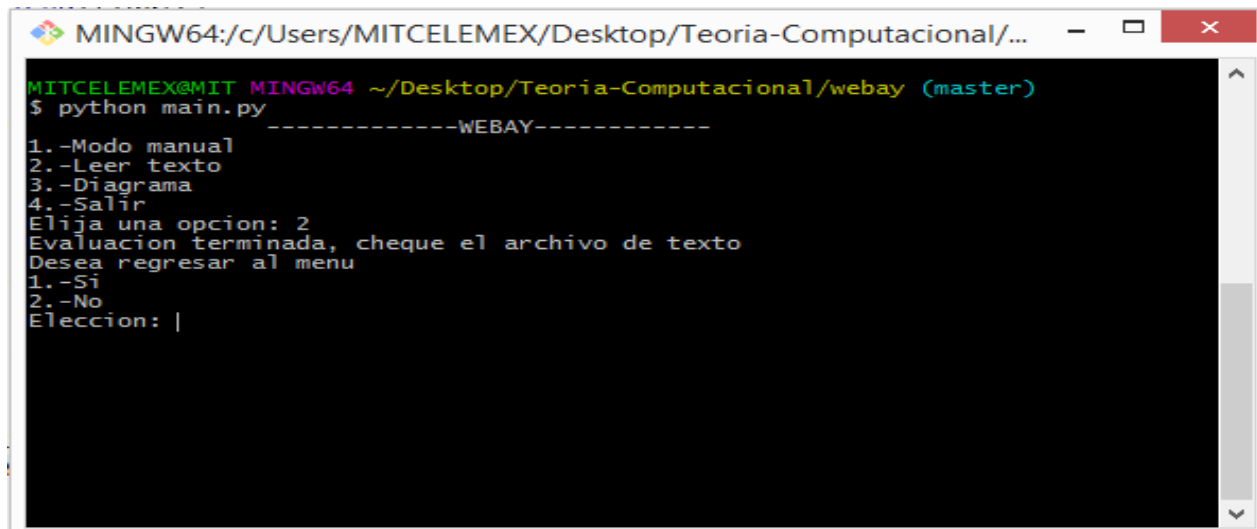
```

q0--w-->q1    q1--e-->q2    q2--b-->q3    q3-- -->q0
q0--e-->q4    q4--b-->q5    q5--a-->q6    q6--y-->q7    q7-- -->q0
q0--r-->q0    q0--e-->q4    q4--t-->q0    q0--e-->q4    q4--y-->q0    q0-- -->q0
q0--w-->q1    q1--e-->q2    q2--b-->q3    q3--a-->q6    q6--y-->q7

```

Figura 3: Historia de la evaluación del autómata

Para el modo de lectura de un archivo:



```

MINGW64:/c/Users/MITCELEMEX/Desktop/Teoria-Computacional/...
MITCELEMEX@MIT MINGW64 ~/Desktop/Teoria-Computacional/webay (master)
$ python main.py
-----WEBAY-----
1.-Modo manual
2.-Leer texto
3.-Diagrama
4.-Salir
Elija una opcion: 2
Evaluacion terminada, cheque el archivo de texto
Desea regresar al menu
1.-Si
2.-No
Eleccion: |

```

Figura 4: Lectura de un texto con palabras WEB

```

Palabra: web Numero de palabra: 4 Numero de fila: 1
Palabra: web) Numero de palabra: 9 Numero de fila: 1
Palabra: web Numero de palabra: 18 Numero de fila: 1
Palabra: web Numero de palabra: 46 Numero de fila: 1
Palabra: web Numero de palabra: 5 Numero de fila: 2
Palabra: web Numero de palabra: 4 Numero de fila: 4
Palabra: web Numero de palabra: 29 Numero de fila: 4
Palabra: web Numero de palabra: 48 Numero de fila: 4
Palabra: web Numero de palabra: 64 Numero de fila: 4
Palabra: web Numero de palabra: 79 Numero de fila: 4
Palabra: web Numero de palabra: 2 Numero de fila: 5
Palabra: website. Numero de palabra: 17 Numero de fila: 5
Palabra: website Numero de palabra: 18 Numero de fila: 5
Palabra: websites Numero de palabra: 43 Numero de fila: 5
Palabra: web Numero de palabra: 78 Numero de fila: 5
Palabra: web-like Numero de palabra: 71 Numero de fila: 7
Palabra: "web" Numero de palabra: 109 Numero de fila: 7
Palabra: web Numero de palabra: 91 Numero de fila: 9
Palabra: web Numero de palabra: 99 Numero de fila: 9
Palabra: worldwideweb, Numero de palabra: 101 Numero de fila: 9
Palabra: web:[12] Numero de palabra: 117 Numero de fila: 9
Palabra: web Numero de palabra: 120 Numero de fila: 9

```

Figura 5: Palabras encontradas del texto

q0--t-->q0	q0--h-->q0	q0--e-->q4	q4-- -->q0	q0--w-->q1	q1--o-->q0	q0--r-->q0	q0--l-->q0
q0--d-->q0	q0-- -->q0	q0--w-->q1	q1--i-->q0	q0--d-->q0	q0--e-->q4	q4-- -->q0	q0--w-->q1
q1--e-->q2	q2--b-->q3	q3-- -->q0					
q0--(-->q0	q0--a-->q0	q0--b-->q0	q0--b-->q0	q0--r-->q0	q0--e-->q4	q4--v-->q0	q0--i-->q0
q0--a-->q0	q0--t-->q0	q0--e-->q4	q4--d-->q0	q0-- -->q0	q0--w-->q1	q1--w-->q1	q1--w-->q1
q1-- -->q0	q0--o-->q0	q0--r-->q0	q0-- -->q0	q0--t-->q0	q0--h-->q0	q0--e-->q4	q4-- -->q0
q0--w-->q1	q1--e-->q2	q2--b-->q3	q3--)-->q0	q0-- -->q0			
q0--i-->q0	q0--s-->q0	q0-- -->q0	q0--a-->q0	q0--n-->q0	q0-- -->q0	q0--i-->q0	q0--n-->q0
q0--f-->q0	q0--o-->q0	q0--r-->q0	q0--m-->q0	q0--a-->q0	q0--t-->q0	q0--i-->q0	q0--o-->q0
q0--n-->q0	q0-- -->q0	q0--s-->q0	q0--p-->q0	q0--a-->q0	q0--c-->q0	q0--e-->q4	q4-- -->q0
q0--w-->q1	q1--h-->q0	q0--e-->q4	q4--r-->q0	q0--e-->q4	q4-- -->q0	q0--d-->q0	q0--o-->q0
q0--c-->q0	q0--u-->q0	q0--m-->q0	q0--e-->q4	q4--n-->q0	q0--t-->q0	q0--s-->q0	q0-- -->q0
q0--a-->q0	q0--n-->q0	q0--d-->q0	q0-- -->q0	q0--o-->q0	q0--t-->q0	q0--h-->q0	q0--e-->q4
q4--r-->q0	q0-- -->q0	q0--w-->q1	q1--e-->q2	q2--b-->q3	q3-- -->q0		
q0--r-->q0	q0--e-->q4	q4--s-->q0	q0--o-->q0	q0--u-->q0	q0--r-->q0	q0--c-->q0	q0--e-->q4
q4--s-->q0	q0-- -->q0	q0--a-->q0	q0--r-->q0	q0--e-->q4	q4-- -->q0	q0--i-->q0	q0--d-->q0
q0--e-->q4	q4--n-->q0	q0--t-->q0	q0--i-->q0	q0--f-->q0	q0--i-->q0	q0--e-->q4	q4--d-->q0
q0-- -->q0	q0--b-->q0	q0--y-->q0	q0-- -->q0	q0--u-->q0	q0--n-->q0	q0--i-->q0	q0--f-->q0
q0--o-->q0	q0--r-->q0	q0--m-->q0	q0-- -->q0	q0--r-->q0	q0--e-->q4	q4--s-->q0	q0--o-->q0
q0--u-->q0	q0--r-->q0	q0--c-->q0	q0--e-->q4	q4-- -->q0	q0--l-->q0	q0--o-->q0	q0--c-->q0
q0--a-->q0	q0--t-->q0	q0--o-->q0	q0--r-->q0	q0--s-->q0	q0-- -->q0	q0--(-->q0	q0--u-->q0

Figura 6: Historia de la evaluación del autómata

2. Balanceo de Paréntesis

2.1. Descripción del programa

El siguiente programa verificar que los paréntesis de una cadena esten balanceados, el programa tiene un modo manual, la entrada de este modo acepta todo tipo de caracteres ASCII. De igual forma cuenta con un modo automático con el cual se genera una cadena de paréntesis aleatoria con una longitud que puede estar entre 0 y 1000.

En un archivo se muestra la historia que se siguió al momento de hacer la evaluación de los paréntesis, así como las reglas que se utilizaron al momento de evaluar cada carácter.

2.2. Código

El código utilizado para la resolución del problema se muestra a continuación:

Código:balanceo.py

```
def VerificarBalanceo(cadena, archivo):
    exp= 'B'
    i=-1;
    cadena_aux=''
    archivo.write(exp)
    while True:
        try:
            i=i+1
            archivo.write('\n')
            if(cadena[i]=='('):
                cadena_aux=cadena_aux+'('
                if(exp[0]=='B'):
                    exp=exp.replace('B', 'RB', 1)
                    cadena_aux=cadena_aux.replace('B', 'RB', 1)
                    archivo.write(cadena_aux+exp)
                    archivo.write('\t\t\tB->(RB')
                    continue
                if(exp[0]=='R'):
                    exp=exp.replace('R', 'RR', 1)
                    cadena_aux=cadena_aux.replace('R', 'RR', 1)
                    archivo.write(cadena_aux+exp)
                    archivo.write('\t\t\tR->(RR')
                    continue
            if(cadena[i]==')'):
                if(exp[0]=='R'):
                    exp=exp[1:]
```

```

        cadena_aux=cadena_aux+' '
        archivo.write(cadena_aux+exp)
        archivo.write('\t\t\tR->_')
        continue
    elif(exp[0]== 'B') :
        exp= ''
        print('Cadena_no_balanceada')
        break
    else :
        exp= ''
        continue
except:
    if(exp=='B') :
        archivo.write(cadena_aux)
        archivo.write('\t\t\tB->_e_\n')
        print('Cadena_balanceada')
    else :
        print('Cadena_no_balanceada')
    break

```

Código:main.py

```

import random
import balanceo

def IniciarArchivo():
    archivo=open("Gramatica.txt","w")
    archivo.close

def Menu():
    print("-----Menu-----")
    print("1.-Modo_Manual")
    print("2.-Modo_Automatico")
    print("3.-Salir")

def Eleccion():
    op=input("Elige_una_opcion:_")
    try:
        op=int(op)
    except:
        print("Introduzca_una_opcion_valida")
    return op

def longitud():
    lon=random.randint(1,100)
    return lon

def generarCadena():
    lon=longitud()
    cadena= ''
    for c in range(1,lon+1):
        o=random.randint(1,2)

```

```

        if(o==1):
            cadena=cadena+'('
        else:
            cadena=cadena+')'
    return cadena

def Manual():
    try:
        archivo=open("Gramatica.txt","a")
    except:
        print("Error_al_abrir_el_archivo")
        exit()

    cadena=input("Introduce una cadena de parentesis: ")
    balanceo.VerificarBalanceo(cadena, archivo)

def Automatico():
    try:
        archivo=open("Gramatica.txt","a")
    except:
        print("Error_al_abrir_el_archivo")
        exit()
    cadena=generarCadena()
    print(cadena)
    balanceo.VerificarBalanceo(cadena, archivo)

def VerificarDeNuevo():
    opcion=input("Desea ingresar una nueva cadena [s/n]: ")
    return opcion

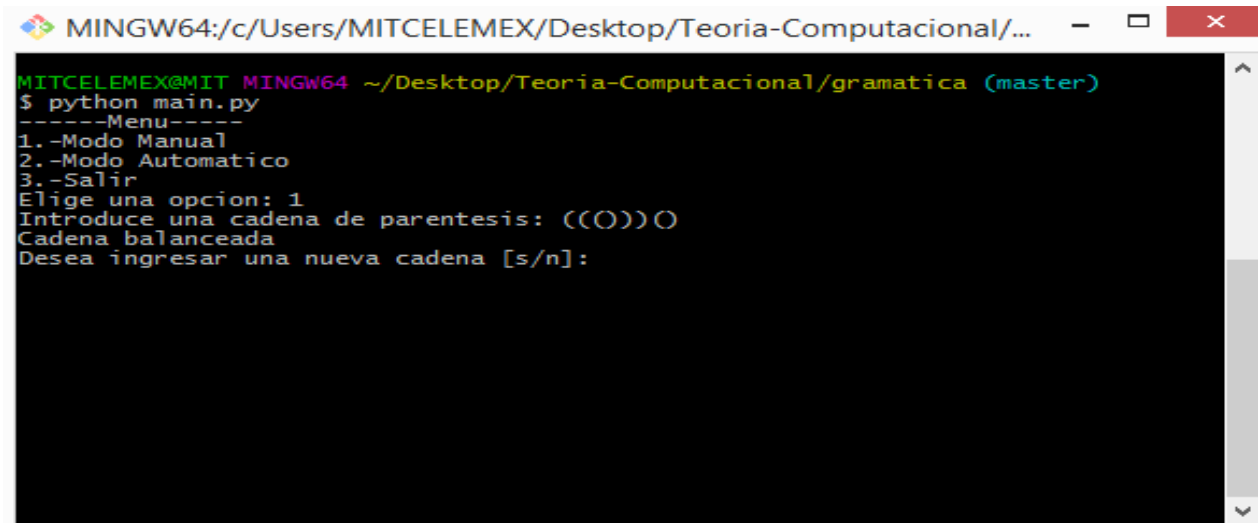
def main():
    IniciarArchivo()
    Menu()
    op=Eleccion()
    while True:
        if(op==1):
            Manual()
            while True:
                rop=VerificarDeNuevo()
                if(rop=='s'):
                    break
                elif(rop=='n'):
                    exit()
                else:
                    continue
            elif(op==2):
                Automatico()
                while True:
                    rop=random.randint(1,2)
                    if(rop==1):
                        break
                    elif(rop==2):
                        exit()

```

```
        else:  
            continue  
elif (op==3):  
    exit ()  
else:  
    continue  
  
main()
```

2.3. Pruebas

A continuación se mostraran algunas imágenes capturadas al momento de ejecutar el programa, dichas imágenes mostraran los resultados obtenidos.
Para el modo manual:



```
MINGW64:/c/Users/MITCELEMEX/Desktop/Teoria-Computacional/...
MITCELEMEX@MIT MINGW64 ~/Desktop/Teoria-Computacional/gramatica (master)
$ python main.py
-----Menu-----
1.-Modo Manual
2.-Modo Automatico
3.-Salir
Elige una opcion: 1
Introduce una cadena de parentesis: ((() ))
Cadena balanceada
Desea ingresar una nueva cadena [s/n]:
```

Figura 7: Cadena de paréntesis de prueba: ((()))0

B		
(RB	B -> (RB	
((RRB	R -> (RR	
(((RRRB	R -> (RR	
((()RRB	R ->)	
((())RB	R ->)	
((()))B	R ->)	
((()))(RB	B -> (RB	
((()))()B	R ->)	
((()))()	B -> e	

Figura 8: Historia de la evaluación de la cadena


```
MITCELEMEX@MIT MINGW64 ~/Desktop/Teoria-Computacional/gramatica (master)
$ python main.py
-----Menu-----
1.-Modo Manual
2.-Modo Automatico
3.-Salir
Elige una opcion: 2
))O(((O((O O))O)))((O))O((O(O O O)))((O))O((O))O(((O O))
(O
Cadena no balanceada
(((O))(((O O((O O))O))O))O O O((O O))O))O((O))O))O(((
O)))O)O
Cadena no balanceada
))O))O))
Cadena no balanceada
O O))O O))O((O((O O)))((O))O)))((O(O(O(O O O O O))O))O((((
(O(O O(
Cadena no balanceada
```

[illegible]

16

3. Palindromos-Gramática libre de contexto

3.1. Descripción del programa

El siguiente programa genera palindromos de cadenas binarias, esto se logra gracias a las siguientes reglas de producción de la gramática:

- 1.-S→e
- 2.-S→0
- 3.-S→1
- 4.-S→0S0
- 5.-S→1S1

El programa cuenta con un modo manual y automático, con el cual se podrá elegir el tamaño del palindromo, como salidas se obtendrá en un archivo la forma en que se fue generando y en otro archivo la regla que se utilizó en cada caso.

3.2. Código

El código utilizado para la resolución del problema se muestra a continuación:

Código:palindromo.py

```
import random;

def IniciarArchivo():
    archivo=open("palindromo.txt","w")
    archivo.close
    historia=open("historiaPal.txt","w")
    historia.close

def Run(pal,longitud,par):
    try:
        archivo=open("palindromo.txt","a")
        historia=open("historiaPal.txt","a")
    except:
        exit()
    archivo.write("S")
    exito=generar_palindromo1(archivo,pal,longitud,par,historia)
    archivo.write("\n\n")
    historia.write("————\n\n")
    archivo.close
    historia.close
    return exito

def opcion_fin(par):
```

```

        if (par==True) :
            op=1
        else :
            op=random.randint(2,3)
        return op

def opcion_inicio() :
    op=random.randint(4,5)
    return op

def regla1(pal, historia) :
    pal=pal.replace("S", "")
    historia.write("\n1.-S-->e")
    return pal

def regla2(pal, historia) :
    pal=pal.replace("S", '0')
    historia.write("\n2.-S-->0")
    return pal

def regla3(pal, historia) :
    pal=pal.replace("S", '1')
    historia.write("\n3.-S-->1")
    return pal

def regla4(pal, historia) :
    pal=pal.replace("S", "0S0")
    historia.write("\n4.-S-->0S0")
    return pal

def regla5(pal, historia) :
    pal=pal.replace("S", "1S1")
    historia.write("\n5.-S-->1S1")
    return pal

def generar_palindromo1(archivo, pal, longitud, par, historia) :

    if (longitud>1):

        opcion=opcion_inicio()
        if (opcion==4):
            pal=regla4(pal, historia)
            archivo.write("\n"+pal)
        if (opcion==5):
            pal=regla5(pal, historia)
            archivo.write("\n"+pal)

    if (longitud==1):
        opcion=opcion_fin(par)
        if (opcion==1):
            pal=regla1(pal, historia)
            archivo.write("\n"+pal)
        if (opcion==2):

```

```

        pal=regla2(pal, historia)
        archivo.write("\n"+pal)
    if(opcion==3):
        pal=regla3(pal, historia)
        archivo.write("\n"+pal)
    if (longitud==0):
        return 1
    longitud=longitud-1
    pal=generar_palindromo1(archivo, pal, longitud, par, historia)

```

Código:main.py

```

import palindromo
import random

def longitud():
    x=random.randint(0,1000);
    return x

palindromo.IniciarArchivo()
def menu():
    print("——Menu——")
    print("1.-Modo_Manual")
    print("2.-Modo_Automatico")
    print("3.-Salir")

while True:
    menu()
    opcion=input("Seleccione_una_opcion:_")
    if(opcion=="1"):
        tamaño=input("Ingrese_un_tamaño_de_cadena:_")
        print(tamaño)
        tamaño=int(tamaño)
        if(tamaño%2==0):
            g=palindromo.Run("S", (tamaño/2)+1, True)
        else:
            tamaño=int(tamaño/2)+1
            g=palindromo.Run("S", tamaño, False)

        while True:
            try:
                seln=input("Desea_regresar_al_menu\n1.-Si\n2.-No\n_")
                seln=int(seln)
            except:
                exit()

            if(seln==1):
                break

```

```

        elif (seln==2):
            exit ()
        else :
            continue
elif (opcion=="2") :
    tamaño=longitud ()
    print (tamaño)
    if (tamaño%2==0):
        g=palindromo.Run("S" ,(tamaño/2)+1,True)
    else :
        tamaño=int (tamaño/2)+1
        g=palindromo.Run("S" ,tamaño, False)

while True:
    try:
        seln=input("Desea_regresar_al_menu\n1.- Si\n2.- No\n-")
        seln=int(seln)
    except:
        exit ()

    if (seln==1):
        break
    elif (seln==2):
        exit ()
    else :
        continue

elif (opcion=="3") :
    exit ()

else :
    print ("Seleccione_una_opcion_correcta")

```

3.3. Pruebas

A continuación se mostraran algunas imágenes capturadas al momento de ejecutar el programa, dichas imágenes mostraran los resultados obtenidos.
Para el modo manual:

```
MITCELEMEX@MIT MINGW64 ~/Desktop/Teoria-Computacional/Palindromo (master)
$ python main.py
-----Menu-----
1.-Modo Manual
2.-Modo Automatico
3.-Salir
Seleccione una opcion: 1
Ingrese un tamaño de cadena: 5
5
11011
Desea regresar al menu
1.-Si
2.-No
- |
```

Figura 11: El tamaño de la cadena sera de 5

```
5
151
11511
11011
```

Figura 12: Evaluación de la gramática

```
5.-S-->151
5.-S-->151
2.-S-->0-----
```

Figura 13: Historia de la evaluación de la gramática

Para el modo automático:

```
MITCELEMEX@MIT MINGW64 ~/Desktop/Teoria-Computacional/Palindromo (master)
$ python main.py
-----Menu-----
1.-Modo Manual
2.-Modo Automatico
3.-Salir
Seleccione una opcion: 2
413
10110100001000101011111000001001101111101011001000000011100001000101111111001111
11101001001111111011110001001011110100111001110111001100010110011111000010101101
00101101100110000000110001010110111011110000000000000111101110110101000110000000
11001101101001011010100001111100110100011001110111001110010111101001000111101111
1110010010111111100111111010001000011100000001001101011111011001000001111101010
0010000101101
Desea regresar al menu
1.-Si
2.-No
- 2
```

Figura 14: Tamaño generado automáticamente

```

5
151
10501
1015101
101151101
10110501101
1011015101101
101101050101101
10110100500101101
1011010005000101101
10110100001510000101101
1011010000105010000101101
101101000010050010000101101
10110100001000500010000101101
1011010000100015100010000101101
101101000010001050100010000101101
10110100001000101510100010000101101
1011010000100010105010100010000101101
101101000010001010151010100010000101101
10110100001000101011511010100010000101101
10110100001000101011115111010100010000101101
1011010000100010101111151111010100010000101101
101101000010001010111110501111010100010000101101
1011010000100010101111005001111010100010000101101
101101000010001010111100050001111010100010000101101
10110100001000101011110000500001111010100010000101101
1011010000100010101111000005000001111010100010000101101
101101000010001010111100000151000001111010100010000101101
10110100001000101011110000010501000001111010100010000101101
1011010000100010101111000001005001000001111010100010000101101
101101000010001010111100000100151001000001111010100010000101101
10110100001000101011110000010011511001000001111010100010000101101
1011010000100010101111000001001105011001000001111010100010000101101
101101000010001010111100000100110151011001000001111010100010000101101
10110100001000101011110000010011011511011001000001111010100010000101101

```

Figura 15: Historia de la evaluación del modo automático

```

|
5.-S-->151
4.-S-->050
5.-S-->151
5.-S-->151
4.-S-->050
5.-S-->151
4.-S-->050
4.-S-->050
4.-S-->050
4.-S-->050
5.-S-->151
4.-S-->050
4.-S-->050
4.-S-->050
5.-S-->151
4.-S-->050
5.-S-->151
5.-S-->151
5.-S-->151

```

Figura 16: Historia de la evaluación de la gramática