INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

UNIDAD DE APRENDIZAJE

TEORÍA COMPUTACIONAL

TAREA

REPORTES DE PRÁCTICAS

ALUMNO

BARRERA ESTRELLA EMANUEL

PROFESOR

JUÁREZ MARTÍNEZ GENARO

GRUPO

2CM1

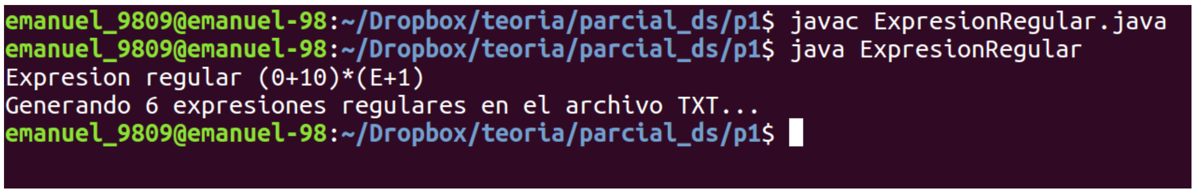
1. Expresión Regular

Este programa que se presenta genera automáticamente cadenas de la expresión regular $(0+10)^{*}(\varepsilon+1)$, se codificó en lenguaje Java.

```java
import java.io.*;
import java.util.Random;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ExpresionRegular {
    public static void main(String[] args) {
        String cadena;
        int primer_union, veces, segunda_union;
        String union1, union2;
        Random r= new Random();
        comand("rm expresion_regular.txt");
        comand("echo >> expresion_regular.txt");
        PrintWriter writer;
            try {
                writer = new PrintWriter("
                    expresion_regular.txt", "UTF-8");
                System.out.println("Expresion regular 
                    (0+10)*(E+1)");
                System.out.println("Generando 6 
                    expresiones regulares en el archivo 
                    TXT...");
                for(int i=0;i<6;i++){
                    cadena="";
                    union1="";
                    union2="";
                    primer_union = r.nextInt(2);
                    veces=r.nextInt(100);
                    switch(primer_union){
                        case 0:
                            union1="10";
                            break;
                        case 1:
                            union1="0";
                            break;
                    }
                    writer.println("De la union de 0+10 
                        se ha usado "+union1);
                    writer.println("Se repite "+veces+" 
                        veces");
                    for(int j=0;j<veces;j++){
                        cadena+=union1;
                    }
                    segunda_union= r.nextInt(2);
                    switch(segunda_union){
                        case 0:
                            union2="E";
                            break;
                        case 1:
                            union2="1";
                            break;
                    }
                    writer.println("De␣la␣segunda␣union␣de␣E+1␣se␣ha␣usado␣"+union2);
                    cadena+=union2;
                    writer.println("La␣cadena␣"+(i+1)+"␣es:␣"+cadena);
                    writer.println("***********************************************");
                }
            writer.close();
        } catch (FileNotFoundException ex) {
            Logger.getLogger(ExpresionRegular.class.getName()).log(Level.SEVERE, null, ex);
        } catch (UnsupportedEncodingException ex) {
            Logger.getLogger(ExpresionRegular.class.getName()).log(Level.SEVERE, null, ex);
        }


    }

    public static char random_01() {
        char letra = 0;
        Random rand = new Random();
        if (rand.nextInt(2) == 0) {
            letra = '0';
        } else {
            letra = '1';
        }
        return letra;
    }

    public static void comand(String cmd) {
        try {
            Process p = Runtime.getRuntime().exec(cmd);
            BufferedReader stdInput
```

```
                              = new BufferedReader(new
                                  InputStreamReader(p.getInputStream
                                  ()));
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }

        public static String scan() {
            String scan = "";
            char c = 0;
            BufferedReader br
                    = new BufferedReader(new
                        InputStreamReader(System.in));
            while (c != '\n') {
                try {
                    c = (char) br.read();
                    if (c != '\n') {
                        scan += c;
                    }
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
            return scan;
        }
    }
```

Su ejecución es:



```
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p1$ javac ExpresionRegular.java
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p1$ java ExpresionRegular
Expresion regular (0+10)*(E+1)
Generando 6 expresiones regulares en el archivo TXT...
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p1$
```

Su salida es:

```
De la union de 0+10 se ha usado 0
Se repite 23 veces
De la segunda union de E+1 se ha usado 1
La cadena 1 es: 000000000000000000000001
*************************************************
De la union de 0+10 se ha usado 10
Se repite 64 veces
De la segunda union de E+1 se ha usado 1
```

```
La cadena 2 es:
    10101010101010101010101010101010101010101010101010101010101010101010101010101

********************************************
De la union de 0+10 se ha usado 0
Se repite 83 veces
De la segunda union de E+1 se ha usado E
La cadena 3 es:
    00000000000000000000000000000000000000000000000000000000000000000000000000000
    E
********************************************
De la union de 0+10 se ha usado 0
Se repite 91 veces
De la segunda union de E+1 se ha usado E
La cadena 4 es:
    00000000000000000000000000000000000000000000000000000000000000000000000000000
    E
********************************************
De la union de 0+10 se ha usado 0
Se repite 59 veces
De la segunda union de E+1 se ha usado 1
La cadena 5 es:
    000000000000000000000000000000000000000000000000000000000001

********************************************
De la union de 0+10 se ha usado 0
Se repite 73 veces
De la segunda union de E+1 se ha usado E
La cadena 6 es:
    0000000000000000000000000000000000000000000000000000000000000000000000000
    E
********************************************
```

2. Autómata PDA

El programa siguiente es un autómata que verifica si una cadena pertene al lenguaje $\{w \mid 0^n 1^n\}$ y genera una animación de cómo funciona usando un conjunto de estados y una pila.

```java
import java.io.*;
import java.util.Random;

public class PushdownAutomata {

    public static final String ANSI_RESET = "\u001B[0m";
    public static final String ANSI_BLACK = "\u001B[30m";
    public static final String ANSI_RED = "\u001B[31m";
    public static final String ANSI_GREEN = "\u001B[32m";
    public static final String ANSI_YELLOW = "\u001B[33m";
    public static final String ANSI_BLUE = "\u001B[34m";
    public static final String ANSI_PURPLE = "\u001B[35m";
    public static final String ANSI_CYAN = "\u001B[36m";
    public static final String ANSI_WHITE = "\u001B[37m";

    public static void main(String[] args) {
        String cadena = "";
        PDA pda = new PDA();
        String s_opc;
        int opc = 0;
        do {
            System.out.println("PDA");
            System.out.println("1)Generar␣cadena␣de␣manera␣manual");
            System.out.println("2)Generar␣cadena␣de␣manera␣aleatoria");
            System.out.println("3)Salir");
            System.out.println("Ingrese␣una␣opcion");
            s_opc = scan();
            try {
                opc = Integer.parseInt(s_opc);
            } catch (NumberFormatException ex) {
                opc = 0;
            }
        } while (!(opc >= 1 && opc <= 3));
        if (opc >= 1 && opc <= 2) {
            comand("rm␣estados.txt");
            comand("echo␣>>␣estados.txt");
            try {
                PrintWriter writer
                        = new PrintWriter("estados.txt", "UTF-8");
                writer.print("{␣i,␣");

                if (opc == 1) {
                    char c = 0;
                    BufferedReader br
                            = new BufferedReader(new InputStreamReader(System.in));
                    while (c != '\n') {
                        try {
                            c = (char) br.read();
            if (c != '\n') {
                cadena += c;
                pda.analizar(c);
                if (pda.getEstado() == 'f') {
                    writer.print("p,␣");
                } else {
                    writer.print("" + pda.getEstado() + ",␣");
                }
            }
        } catch (java.io.IOException ex) {
            ex.printStackTrace();
        }
    }
}
if (opc == 2) {
    char c = 0;
    Random rand = new Random();
    int random = rand.nextInt(4);
    if (random == 0) {
        random = rand.nextInt(100);
        for (int i = 0; i < random; i++) {
            c = random_01();
            cadena += c;
            pda.analizar(c);
            if (pda.getEstado() == 'f') {
                writer.print("p,␣");
            } else {
                writer.print("" + pda.getEstado() + ",␣");
            }
            System.out.print(c);
        }
        System.out.println();
    } else {
        random = rand.nextInt(50);
        for (int i = 0; i < random; i++) {
            c = '0';
            cadena += c;
            pda.analizar(c);
            if (pda.getEstado() == 'f') {
```

```java
                        writer.print("p, ");
                    } else {
                        writer.print("" + pda.getEstado() + ", ");
                    }
                    System.out.print(c);
                }
                for (int i = 0; i < random; i++) {
                    c = '1';
                    cadena += c;
                    pda.analizar(c);
                    if (pda.getEstado() == 'f') {
                        writer.print("p, ");
                    } else {
                        writer.print("" + pda.getEstado() + ", ");
                    }
                    System.out.print(c);
                }
                System.out.println();
            }
        }
        writer.println("f }");
        writer.close();
        if (pda.getEstado() == 'f') {
            System.out.println(ANSI_GREEN + "La cadena pertenece al lenguaje" + ANSI_GREEN);

            AnimacionPda animacion = new AnimacionPda();
            animacion.animar(cadena, 0);
            animacion.setVisible(true);
            esperar(1f);
            int i = 0;
            do {
                animacion.animar(cadena.substring(i + 1), (i + 1));
                animacion.setVisible(true);
                esperar(1f);
                i++;
            } while (cadena.charAt(i) == '0');
            cadena = cadena.substring(i);
            for (int j = 0; j < cadena.length(); j++) {
```

```
                        i--;
                        animacion.animar(cadena.substring(j + 1), i);
                        animacion.setVisible(true);
                        esperar(1f);
                    }
                } else {
                    System.out.println(ANSI_RED + "La␣cadena␣no␣pertenece␣al␣lenguaje" + ANSI_RED);
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }

    public static char random_01() {
        Random rand = new Random();
        return ((char) (rand.nextInt(2) + 48));
    }

    public static void comand(String cmd) {
        try {
            Process p = Runtime.getRuntime().exec(cmd);
            BufferedReader stdInput
                    = new BufferedReader(new InputStreamReader(p.getInputStream()));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public static String scan() {
        String scan = "";
        char c = 0;
        BufferedReader br
                = new BufferedReader(new InputStreamReader(System.in));
        while (c != '\n') {
            try {
                c = (char) br.read();
                if (c != '\n') {
                    scan += c;
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
        return scan;
    }

    public static void esperar(float s) {
        try {
            Thread.sleep((int) (s * 1000));
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }
}
```

Para la animación se usó:

```
import java.awt.*;
import javax.swing.*;

public class PumppingLemma extends JPanel {

    private int n;
    private String cadena;
    private int x_rect;
    private int y_rect;
    private int y_space;
    private int x_space;

    public PumppingLemma(String s,int i) {
        cadena=s;
        n = i;
        x_rect = 40;
        y_rect = 30;
        y_space = 80;
        x_space = 120;
    }

    public void setCadena(String cadena) {
        this.cadena = cadena;
    }


    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
```

```java
        Graphics2D g2 = (Graphics2D) g;
        g2.setColor(Color.BLACK);
        g2.drawString(cadena, x_space + 70, y_space);
        g2.drawLine(x_space+72, y_space+2, x_space+72,
            y_space+20);
        int[] vx2 = {x_space+72, x_space+76, x_space+68};
        int[] vy2 = {y_space+30, y_space+20, y_space+20};
        g2.fillPolygon(vx2, vy2, 3);

        for (int i = 0; i < n; i++) {

            //Contorno del espacio de la pila
            g2.setColor(Color.BLACK);
            g2.drawRect(x_space + 50, (i + 1) * y_rect +
                y_space, x_rect, y_rect);
            g2.setColor(Color.GREEN);
            g2.fillRect(x_space + 51, (i + 1) * y_rect +
                1 + y_space, x_rect - 1, y_rect - 1);
            g2.setColor(Color.BLACK);
            g2.drawString("X",x_space + 50+17, (i + 1) *
                y_rect + y_space+20);

        }
        g2.setColor(Color.BLACK);
        g2.drawRect(x_space + 50, (n + 1) * y_rect +
            y_space, x_rect, y_rect);
        g2.setColor(Color.ORANGE);
        g2.fillRect(x_space + 51, (n + 1) * y_rect + 1 +
            y_space, x_rect - 1, y_rect - 1);
        g2.setColor(Color.BLACK);
        g2.drawString("Z0",x_space + 50+13, (n+ 1) *
            y_rect + y_space+20);
    }
}
```

```java
import java.awt.*;
import java.awt.event.AdjustmentEvent;
import java.awt.event.AdjustmentListener;
import javax.swing.*;

public class AnimacionPda extends JFrame implements
    AdjustmentListener {

    private JScrollPane scroll;
    private JPanel panel;

    public AnimacionPda() {
        setTitle("Animacion");
        setLocation(200, 50);
        setSize(400, 500);
        setDefaultCloseOperation(WindowConstants.
            EXIT_ON_CLOSE);
        setLayout(null);
        scroll = new JScrollPane();
        scroll.setBounds(0, 0, getWidth(), getHeight());
        scroll.setHorizontalScrollBarPolicy(JScrollPane.
            HORIZONTAL_SCROLLBAR_NEVER);
        scroll.getVerticalScrollBar().
            addAdjustmentListener(this);
        add(scroll);
    }

    public void animar(String s, int n) {
        panel = new PumppingLemma(s,n);
        panel.setPreferredSize(new Dimension(400, (n+2)
            *40));
        scroll.setViewportView(panel);
    }

    @Override
    public void adjustmentValueChanged(AdjustmentEvent ae
        ) {
        //animar();
        setVisible(true);
    }
}
```

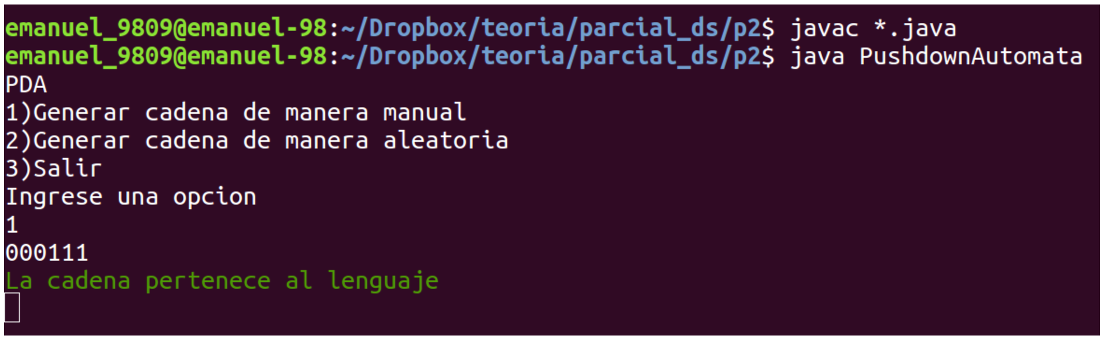
El autómata se codificó de la siguiente manera:

```
public class PDA {

    private Pila pila;
    private char estado;

    public PDA() {
        pila = new Pila();
        estado = 'q';
        pila.push("Z0");
    }

    public char getEstado() {
        return estado;
    }
    public void analizar(char input) {
        if(estado!='e'){
            switch (input) {
                case '0':
                    switch(estado){
                        case 'q':
                            pila.push("X");
                            break;
                        default:
                            estado='e';
                            break;
                    }
                    break;
                case '1':
                    switch(estado){
                        case 'q':
                            estado='p';
                            pila.pop();
                            break;
                        case 'p':
                            pila.pop();
                            break;
                        default:
                            estado='e';
                            break;
                    }
                    break;
                default:
                    estado='e';
            }
            if(pila.top().equals("Z0") && estado!='e'){
                estado='f';
            }
        }
    }
}
```

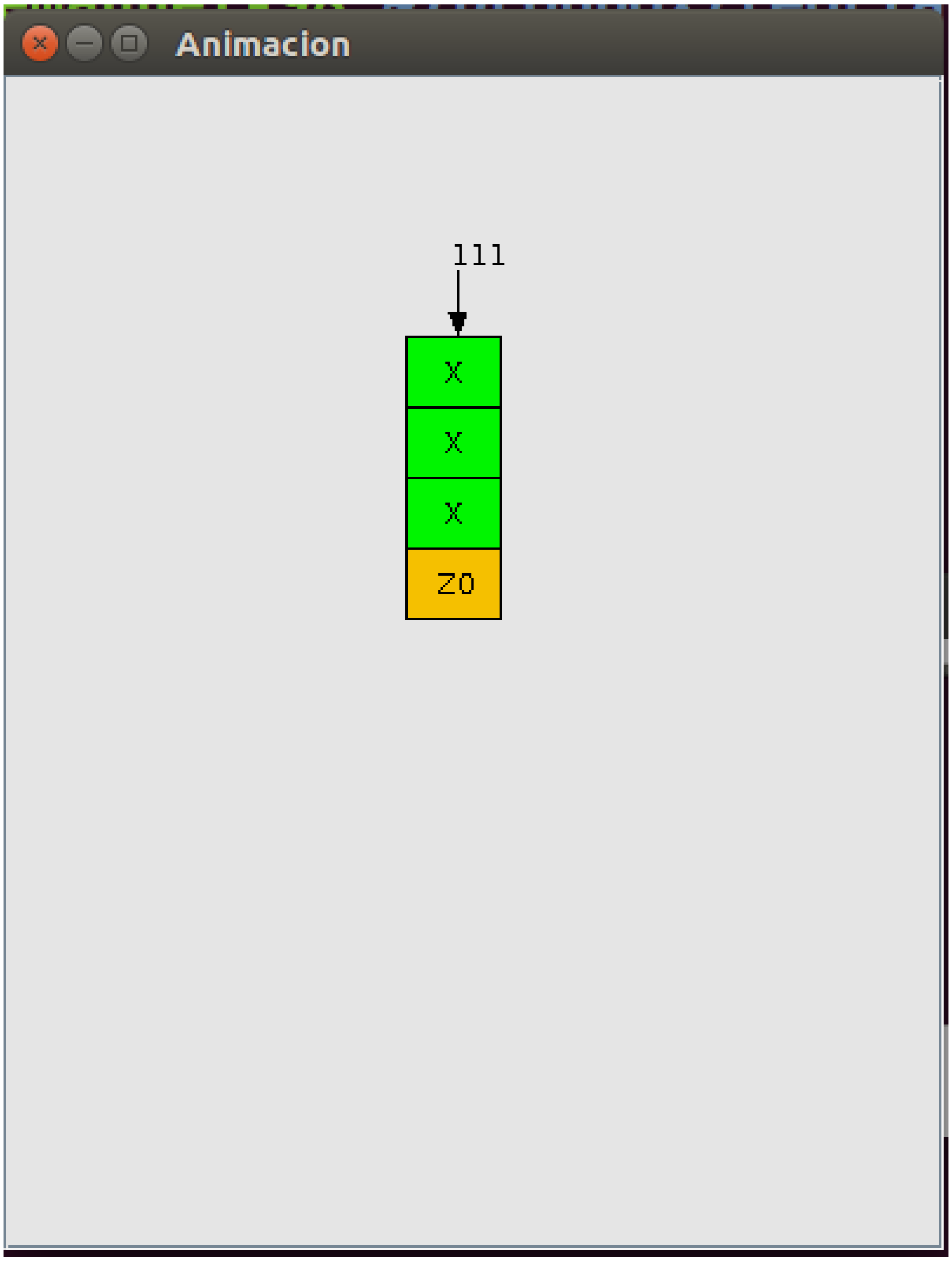
Para el TAD pila se usó:

```
import java.util.ArrayList;

public class Pila {
    private ArrayList<String> pila;
    private int tope;
    public Pila(){
        pila= new ArrayList<>();
        tope=-1;
    }
    public void push(String n){
        pila.add(++tope, n);
    }
    public String pop(){
        String n="";
        if(!estaVacia()){
            n=pila.get(tope);
            pila.remove(tope--);
        }
        return n;
    }
    public String top(){
        String n="";
        if(!estaVacia()){
            n=pila.get(tope);
        }
        return n;
    }
    public boolean estaVacia(){
        return tope==-1;
    }
}
```

Su ejecución es:



```
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p2$ javac *.java
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p2$ java PushdownAutomata
PDA
1)Generar cadena de manera manual
2)Generar cadena de manera aleatoria
3)Salir
Ingrese una opcion
1
000111
La cadena pertenece al lenguaje
```


Animacion
111
X
X
X
Z0

Su salida es:

```
{ i, q, q, q, p, p, p, f }
```

3. Palíndromo

Este programa simula una gramática libre de contexto que genera una cadena que es palíndromo. La gramática se define como:

$$P \to aPa \,|\, b \,|\, \varepsilon$$

```java
import java.io.*;
import java.util.Random;

public class Palindromo {

    public static final String ANSI_RESET = "\u001B[0m";
    public static final String ANSI_BLACK = "\u001B[30m";
    public static final String ANSI_RED = "\u001B[31m";
    public static final String ANSI_GREEN = "\u001B[32m";
    public static final String ANSI_YELLOW = "\u001B[33m"
        ;
    public static final String ANSI_BLUE = "\u001B[34m";
    public static final String ANSI_PURPLE = "\u001B[35m"
        ;
    public static final String ANSI_CYAN = "\u001B[36m";
    public static final String ANSI_WHITE = "\u001B[37m";

    public static void main(String[] args) {
        char c;
        String cadena = "";
        String s_opc;
        int opc = 0;
        do {
            System.out.println("Palindromo");
            System.out.println("P-> aPa | b | E");
            System.out.println("1) Verificar si una cadena
                 cualquiera es palindromo");
            System.out.println("2) Generar palindromo de
                manera manual");
            System.out.println("3) Generar palindromo de
                manera cualquiera");
            System.out.println("4) Salir");
            System.out.println("Ingrese una opcion");
            s_opc = scan();
            try {
                opc = Integer.parseInt(s_opc);
            } catch (NumberFormatException ex) {
                opc = 0;
            }
        } while (!(opc >= 1 && opc <= 4));
        if (opc >= 1 && opc <= 3) {
            comand("rm palindromo.txt");
            comand("echo >> palindromo.txt");
            comand("rm derivaciones.txt");
            comand("echo >> derivaciones.txt");
            try {
                PrintWriter writer_palindromo
                        = new PrintWriter("palindromo.txt
                            ", "UTF-8");
                PrintWriter writer_derivaciones
                        = new PrintWriter("derivaciones.
                            txt", "UTF-8");
                if (opc == 1) {
                    c = 0;
                    int length = 0;
                    boolean esPalindromo = true;
                    boolean epsilon;
                    BufferedReader br
                            = new BufferedReader(new
                                InputStreamReader(System.
                                in));
                    while (c != '\n') {
                        try {
                            c = (char) br.read();
                            if (c != '\n') {
                                cadena += c;
                                length++;
                            }
                        } catch (IOException ex) {
                            ex.printStackTrace();
                        }
                    }
                    for (int i = 0, l = length - 1; i <
                        length; i++, l--) {
                        if (cadena.charAt(i) != cadena.
                            charAt(l)) {
                            esPalindromo = false;
                        }
                    }
                    if (esPalindromo) {
                        if (length % 2 == 0) {
                            epsilon = true;
                        } else {
```

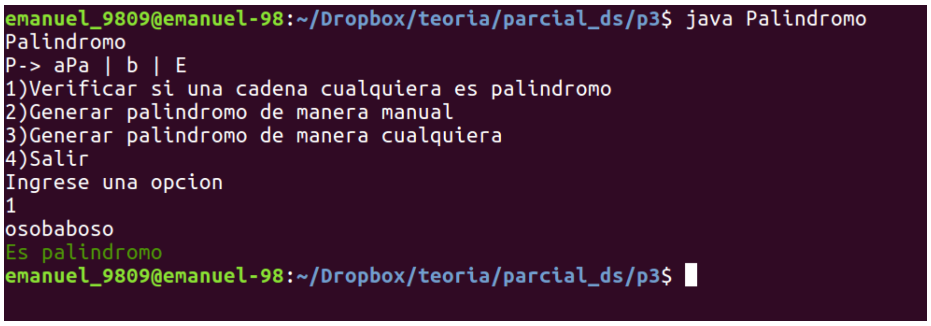
```
                epsilon = false;
            }
            cadena = cadena.substring((length + 1) / 2);
            System.out.println(ANSI_GREEN + "Es␣palindromo" + ANSI_GREEN);
            String palindromo = "";
            if (epsilon) {
                palindromo += 'E' + "";
                writer_palindromo.println("E");
            }
            for (int i = 0; i < cadena.length(); i++) {
                c = cadena.charAt(i);
                switch (palindromo) {
                    case "":
                        if (c != 'E') {
                            writer_derivaciones.println("P␣->␣b");
                        }
                        palindromo = c + "";
                        break;
                    case "E":
                        writer_derivaciones.println("P␣->␣E");
                        palindromo = c + "" + c;
                        writer_derivaciones.println("P␣->␣aPa");
                        break;
                    default:
                        writer_derivaciones.println("P␣->␣aPa");
                        palindromo = c + palindromo + c;
                        break;
                }
                writer_palindromo.println(palindromo);
            }
        } else {
            cadena = "";
```

```
                System.out.println(ANSI_RED + "No␣es␣palindromo" + ANSI_RED);
            }
        }
        if (opc == 2) {
            c = 0;
            BufferedReader br
                    = new BufferedReader(new InputStreamReader(System.in));
            while (c != '\n') {
                try {
                    c = (char) br.read();
                    if (c != '\n') {
                        switch (cadena) {
                            case "":
                                if (c != 'E') {
                                    writer_derivaciones.println("P␣->␣b");
                                }
                                cadena = c + "";
                                break;
                            case "E":
                                writer_derivaciones.println("P␣->␣E");
                                cadena = c + "" + c;
                                writer_derivaciones.println("P␣->␣aPa");
                                break;
                            default:
                                writer_derivaciones.println("P␣->␣aPa");
                                cadena = c + cadena + c;
                                break;
                        }
                        writer_palindromo.println(cadena);
                    }
                } catch (IOException ex) {
                    ex.printStackTrace();
```

```
                        }
                    }
                }
                if (opc == 3) {
                    Random rand = new Random();
                    int random = rand.nextInt(10);
                    for (int i = 0; i < random; i++) {
                        c = random_alphabet();
                        switch (cadena) {
                            case "":
                                if (c != 'E') {
                                    writer_derivaciones.println("P -> b");
                                }
                                cadena = c + "";
                                break;
                            case "E":
                                writer_derivaciones.println("P -> E");
                                cadena = c + "" + c;
                                writer_derivaciones.println("P -> aPa");
                                break;
                            default:
                                writer_derivaciones.println("P -> aPa");
                                cadena = c + cadena + c;
                                break;
                        }
                        writer_palindromo.println(cadena);
                        System.out.print(c);
                    }
                    System.out.println();
                }
                writer_derivaciones.close();
                writer_palindromo.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }

    public static char random_alphabet() {
        Random rand = new Random();
        return ((char) (rand.nextInt(26) + 97));
```

```java
    }

    public static void comand(String cmd) {
        try {
            Process p = Runtime.getRuntime().exec(cmd);
            BufferedReader stdInput
                    = new BufferedReader(new
                      InputStreamReader(p.getInputStream
                      ()));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }

    public static String scan() {
        String scan = "";
        char c = 0;
        BufferedReader br
                = new BufferedReader(new
                  InputStreamReader(System.in));
        while (c != '\n') {
            try {
                c = (char) br.read();
                if (c != '\n') {
                    scan += c;
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
        return scan;
    }
}
```

Su ejecución es:



```
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p3$ java Palindromo
Palindromo
P-> aPa | b | E
1)Verificar si una cadena cualquiera es palindromo
2)Generar palindromo de manera manual
3)Generar palindromo de manera cualquiera
4)Salir
Ingrese una opcion
1
osobaboso
Es palindromo
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p3$
```

Su primer salida despliega el número de derivaciones y cómo se derivó la gramática para hacer el palíndromo:

```
P -> b
P -> aPa
P -> aPa
P -> aPa
P -> aPa
```

Su segunda salida despliega cómo esta generándose la cadena recursivamente de acuerdo a la gramática:

```
a
bab
obabo
sobabos
osobaboso
```

4. Gramática libre de contexto para balancear parentésis

Lo siguiente es un programa que verifica si una expresión está balanceada de paréntesis, la gramática libre de contexto que verifica esta expresión está dada por:

$$B \to (RB \,|\, \varepsilon$$

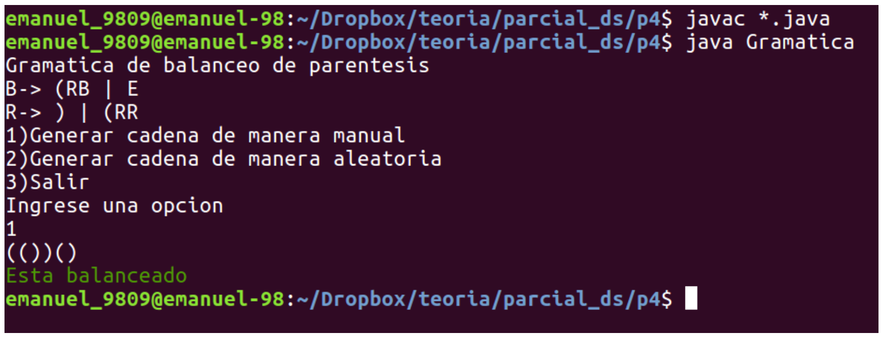
$$R \to) \,|\, (RR$$

```
import java.io.*;
import java.util.Random;

public class Gramatica {

    public static final String ANSI_RESET = "\u001B[0m";
    public static final String ANSI_BLACK = "\u001B[30m";
    public static final String ANSI_RED = "\u001B[31m";
    public static final String ANSI_GREEN = "\u001B[32m";
    public static final String ANSI_YELLOW = "\u001B[33m"
        ;
    public static final String ANSI_BLUE = "\u001B[34m";
    public static final String ANSI_PURPLE = "\u001B[35m"
        ;
    public static final String ANSI_CYAN = "\u001B[36m";
    public static final String ANSI_WHITE = "\u001B[37m";

    public static void main(String[] args) {
        Analizador analizador;
        String s_opc;
        char c;
```

```java
        int opc = 0;
        do {
            System.out.println("Gramatica de balanceo de parentesis");
            System.out.println("B-> (RB | E");
            System.out.println("R-> ) | (RR");
            System.out.println("1)Generar cadena de manera manual");
            System.out.println("2)Generar cadena de manera aleatoria");
            System.out.println("3)Salir");
            System.out.println("Ingrese una opcion");
            s_opc = scan();
            try {
                opc = Integer.parseInt(s_opc);
            } catch (NumberFormatException ex) {
                opc = 0;
            }
        } while (!(opc >= 1 && opc <= 3));
        if (opc >= 1 && opc <= 2) {
            analizador = new Analizador();
            c = 0;
            comand("rm gramatica.txt");
            comand("echo >> gramatica.txt");
            try {
                PrintWriter writer
                        = new PrintWriter("gramatica.txt", "UTF-8");
                writer.println("B");
                if (opc == 1) {
                    BufferedReader br
                            = new BufferedReader(new InputStreamReader(System.in));
                    do {
                        try {
                            c = (char) br.read();
                            analizador.analizar(c);
                            writer.println(analizador.getDerivacion());
                        } catch (IOException ex) {
                            ex.printStackTrace();
                        }
                    }while (c != '\n');
                }
                if (opc == 2) {
                    while (c != '\n') {
                        c = random_parentesis();
                        analizador.analizar(c);
                        System.out.print(c);
                        writer.println(analizador.getDerivacion());
                    }
                    System.out.println();
                }
                if(analizador.isBalanceado()){
                    System.out.println(ANSI_GREEN + "Esta balanceado" + ANSI_GREEN);
                    writer.println("Esta balanceado");
                }else{
                    System.out.println(ANSI_RED + "No esta balanceado" + ANSI_RED);
                    writer.println("No esta balanceado");
                }
                writer.println("");
                writer.close();
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }

    public static char random_parentesis() {
        Random rand = new Random();
        char caracter = 0;
        int random = rand.nextInt(9);
        if (random == 5) {
            caracter = '\n';
        } else {
            caracter = (char) (rand.nextInt(2) + 40);
        }
        return caracter;
    }

    public static void comand(String cmd) {
        try {
            Process p = Runtime.getRuntime().exec(cmd);
            BufferedReader stdInput
                    = new BufferedReader(new InputStreamReader(p.getInputStream()));
        } catch (IOException ex) {
                ex.printStackTrace();
            }
        }

        public static String scan() {
            String scan = "";
            char c = 0;
            BufferedReader br
                    = new BufferedReader(new
                        InputStreamReader(System.in));
            while (c != '\n') {
                try {
                    c = (char) br.read();
                    if (c != '\n') {
                        scan += c;
                    }
                } catch (IOException ex) {
                    ex.printStackTrace();
                }
            }
            return scan;
        }
}
```

```
public class Analizador {
    private boolean balanceado;
    private String derivacion;
    private int index;

    public Analizador(){
        balanceado=true;
        derivacion="B";
        index=0;
    }

    public boolean isBalanceado() {
        return balanceado;
    }

    public String getDerivacion() {
        return derivacion;
    }


    public void analizar(char c) {
        try{
            String derivada="";
            switch(derivacion.charAt(index)){
                case 'B':
                    switch(c){
                        case '(':
                            derivada="(RB";
                            balanceado=false;
                            break;
                        case '\n':
                            derivada="";
                            balanceado=true;
                            break;
                        default:
                            balanceado=false;
                            break;
                    }
                    break;
                case 'R':
                    switch(c){
                        case '(':
                            derivada="(RR";
                            balanceado=false;
                            break;
                        case ')':
                            derivada=")";
                            balanceado=true;
                            break;
                        default:
                            balanceado=false;
                            break;
                    }
                    break;
                default:
                    balanceado=false;
                    break;
            }
            derivacion=derivacion.substring(0, index)+
                    derivada+derivacion.substring(index
                       +1);
            index++;
        }catch(Exception e){
        }
    }
}
```

Su ejecución es:



```
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p4$ javac *.java
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p4$ java Gramatica
Gramatica de balanceo de parentesis
B-> (RB | E
R-> ) | (RR
1)Generar cadena de manera manual
2)Generar cadena de manera aleatoria
3)Salir
Ingrese una opcion
1
(())()
Esta balanceado
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p4$ 
```

Su salida muestra línea por línea las derivaciones que se hicieron para analizar el balanceo de paréntesis:

```
B
(RB
((RRB
(()RB
(())B
(())(RB
(())()B
(())()
Esta balanceado
```

5. Máquina de Turing

Este programa está hecho en base a la regla 110, considerando la máquina $M = \{Q_M, \sum_M, \Gamma, \delta, q_0, B\}$ capaz de emular el comportamiento del autómata celular elemental conocido como Regla 110. $Q_M = \{S_{x0}, S_{01}, S_{11}, S_B\}$ es el conjunto de estados de la máquina, $\sum_M = \{0, 1\}$ es el alfabeto de entrada, $\Gamma = \{0, 1, B\}$ es el alfabeto de la cinta, δ es la función de transición mostrada a continuación:

State	0	1	B
S_{x0}	$S_{x0}, 0, R$	$S_{01}, 1, R$	$S_B, 0, L$
S_{01}	$S_{x0}, 1, R$	$S_{11}, 1, R$	
S_{11}	$S_{x0}, 1, R$	$S_{11}, 0, R$	
S_B	$S_B, 0, L$	$S_B, 1, L$	$S_{x0}, 0, R$

Donde $q_0 = S_{x0}$ es el estado inicial de la máquina y B es el símbolo en blanco. La Máquina de Turing está codificada como:

```java
import java.awt.*;
import javax.swing.*;

public class Turingmachine extends JPanel {

    private int n;
    private int index;
    private String cadena;
    private int x_rect;
    private int y_rect;
    private int y_space;
    private int x_space;

    public Turingmachine(String s, int i) {
        cadena =s;
        n =cadena.length();
        index=i;
        x_rect = 40;
        y_rect = 30;
        y_space = 80;
        x_space = 120;
    }

    public void setCadena(String cadena) {
        this.cadena = cadena;
    }


    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        g2.setColor(Color.BLACK);
        g2.drawRect(x_space + 50+40*(index),  y_rect + y_space, x_rect, y_rect);
        g2.setColor(Color.ORANGE);
        g2.fillRect(x_space + 51+40*(index),  y_rect + 1 + y_space, x_rect - 1, y_rect - 1);
        g2.setColor(Color.BLACK);
        g2.drawString("q",x_space + 50+17+40*(index), y_rect + y_space+20);
        g2.drawLine(x_space+72+40*(index), y_space+60, x_space+72+40*(index), y_space+80);
        g2.fillPolygon(new int[]{x_space+72+40*(index), x_space+76+40*(index), x_space+68+40*(index)},
                new int[] {y_space+90, y_space+80, y_space+80}, 3);
        g2.drawLine(x_space, y_space+90, x_space+n*40+180, y_space+90);
        g2.drawLine(x_space, y_space+120, x_space+n*40+180, y_space+120);
        g2.drawString(".",x_space+40+17, y_rect + y_space+80);
        g2.drawString(".",x_space+50+17, y_rect + y_space+80);
        g2.drawString(".",x_space+60+17, y_rect + y_space+80);
        g2.drawLine(x_space+40+50, y_space+90, x_space+40+50, y_space+120);

        for (int i = 0; i < n; i++) {

            //Contorno del espacio de la pila
            g2.setColor(Color.BLACK);
            g2.drawString(cadena.charAt(i)+"",x_space+40*(i+1)+50+17, y_rect + y_space+80);
            g2.drawLine(x_space+40*(i+2)+50, y_space+90, x_space+40*(i+2)+50, y_space+120);
        }
        g2.drawString(".",x_space+40*(n+2)+17, y_rect + y_space+80);
        g2.drawString(".",x_space+40*(n+2)+27, y_rect + y_space+80);
        g2.drawString(".",x_space+40*(n+2)+37, y_rect + y_space+80);
    }
}
```

```
import java.awt.*;
import java.awt.event.AdjustmentEvent;
import java.awt.event.AdjustmentListener;
import javax.swing.*;

public class AnimacionTM extends JFrame implements AdjustmentListener {

    private JScrollPane scroll;
    private JPanel panel;

    public AnimacionTM() {
        setTitle("Animacion");
        setLocation(200, 50);
        setSize(800, 500);
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        setLayout(null);
        scroll = new JScrollPane();
        scroll.setBounds(0, 0, getWidth(), getHeight());
        scroll.setVerticalScrollBarPolicy(JScrollPane.
                VERTICAL_SCROLLBAR_NEVER);
        scroll.getVerticalScrollBar().
            addAdjustmentListener(this);
        add(scroll);
    }

    public void animar(String s, int i) {
        panel = new Turingmachine(s,i);
        panel.setPreferredSize(new Dimension(s.length() *
            50+200, 500));
        scroll.setViewportView(panel);
    }

    @Override
    public void adjustmentValueChanged(AdjustmentEvent ae
        ) {
        //animar();
        setVisible(true);
    }
}
```

```
public class MaquinaTuring {

    private String estado;
    private char caracter;
    private char movimiento;
    public MaquinaTuring(){
        estado="Sx0";
    }

    public String getEstado() {
        return estado;
    }

    public char getCaracter() {
        return caracter;
    }

    public char getMovimiento() {
        return movimiento;
    }

    public void analizar(char entrada) {
        switch (entrada) {
            case '0':
                switch (estado) {
```

```
                case "Sx0":
                    estado = "Sx0";
                    caracter='0';
                    movimiento='R';
                    break;
                case "S01":
                    estado = "Sx0";
                    caracter='1';
                    movimiento='R';
                    break;
                case "S11":
                    estado = "Sx0";
                    caracter='1';
                    movimiento='R';
                    break;
                case "SB":
                    estado = "SB";
                    caracter='0';
                    movimiento='L';
                    break;
            }
            break;
        case '1':
            switch (estado) {
                case "Sx0":
                    estado = "S01";
                    caracter='1';
                    movimiento='R';
                    break;
                case "S01":
                    estado = "S11";
                    caracter='1';
                    movimiento='R';
                    break;
                case "S11":
                    estado = "S11";
                    caracter='0';
                    movimiento='R';
                    break;
                case "SB":
                    estado = "SB";
                    caracter='1';
                    movimiento='L';
                    break;
            }
            break;
```

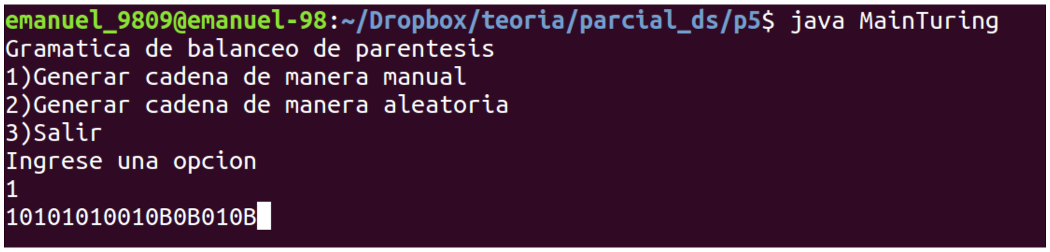
```java
                case 'B':
                    switch (estado) {
                        case "Sx0":
                            estado = "SB";
                            caracter='0';
                            movimiento='L';
                            break;
                        case "SB":
                            estado = "Sx0";
                            caracter='0';
                            movimiento='R';
                            break;
                    }
                    break;
            }
        }

}
```

```java
import java.io.*;
import java.util.Random;

public class MainTuring {
    public static void main(String[] args) {
        AnimacionTM animacion;
        MaquinaTuring maquina;
        String s_opc;
        String cadena;
        String cadena_turing;
        char c;
        int index;
        int opc;
        do {
            System.out.println("Gramatica de balanceo de 
                parentesis");
            System.out.println("1)Generar cadena de 
                manera manual");
            System.out.println("2)Generar cadena de 
                manera aleatoria");
            System.out.println("3)Salir");
            System.out.println("Ingrese una opcion");
            s_opc = scan();
            try {
                opc = Integer.parseInt(s_opc);
            } catch (NumberFormatException ex) {
                opc = 0;
            }
        } while (!(opc >= 1 && opc <= 3));
        if (opc >= 1 && opc <= 2) {
            cadena_turing="";
            cadena="";
            animacion = new AnimacionTM();
            index=1;
            animacion.animar("B",index);
            animacion.setVisible(true);
            maquina = new MaquinaTuring();
            c = 0;
            comand("rm␣turing.txt");
            comand("echo␣>>␣turing.txt");
            try {
                PrintWriter writer
                        = new PrintWriter("turing.txt", "UTF-8");
                writer.print("{␣Sx0␣");
                if (opc == 1) {
                    BufferedReader br
                            = new BufferedReader(new InputStreamReader(System.in));
                    while (c != '\n') {
                        try {
                            c = (char) br.read();
                            cadena+=c+"";
                            maquina.analizar(c);
                            cadena_turing+=maquina.getCaracter()+"";
                            if(maquina.getMovimiento()=='R'){
                                index++;
                            }else if(maquina.getMovimiento()=='L'){
                                index--;
                            }
                            writer.print(maquina.getEstado()+"␣");
                            animacion.animar(cadena,index-1);
                            animacion.setVisible(true);
                            esperar(1f);
                        } catch (IOException ex) {
                            ex.printStackTrace();
                        }
```

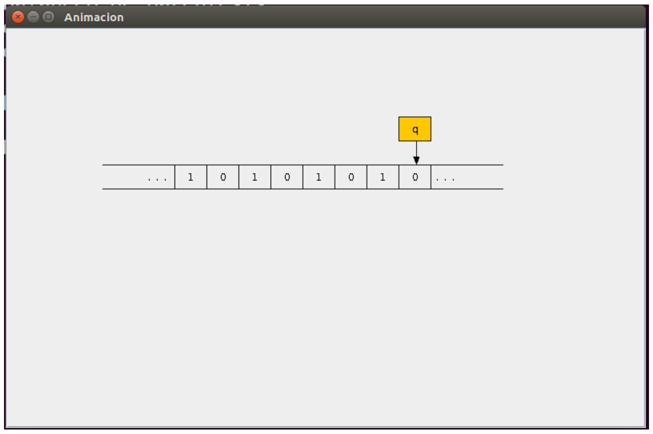
```java
                }
                animacion.setVisible(false);
            }
            if (opc == 2) {
                while (c != '\n') {
                    c = random_01B();
                    System.out.print(c);
                    cadena+=c+"";
                    maquina.analizar(c);
                    cadena_turing+=maquina.getCaracter()+"";
                    if(maquina.getMovimiento()=='R'){
                        index++;
                    }else if(maquina.getMovimiento()=='L'){
                        index--;
                    }
                    writer.print(maquina.getEstado()+" ");
                    animacion.animar(cadena,index-1);
                    animacion.setVisible(true);
                    esperar(1f);
                }
                System.out.println();
                animacion.setVisible(false);
            }
            writer.println("}");
            writer.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
    System.exit(0);
}

public static char random_01B() {
    Random rand = new Random();
    char caracter;
    int random=rand.nextInt(10);
    if(random==9){
        caracter='\n';
    }else if (random<5){
        caracter='B';
    }else{
        caracter=(char) (rand.nextInt(2) + 48);
    }
```

```
        return caracter;
    }

    public static void comand(String cmd) {
        try {
            Process p = Runtime.getRuntime().exec(cmd);
            BufferedReader stdInput
                    = new BufferedReader(new
                        InputStreamReader(p.getInputStream
                        ()));
        } catch (java.io.IOException ex) {
            ex.printStackTrace();
        }
    }

    public static String scan() {
        String scan = "";
        char c = 0;
        BufferedReader br
                = new BufferedReader(new
                    InputStreamReader(System.in));
        while (c != '\n') {
            try {
                c = (char) br.read();
                if (c != '\n') {
                    scan += c;
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
        return scan;
    }

    public static void esperar(float s) {
        try {
            Thread.sleep((int) (s * 1000));
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }
    }
}
```

Su ejecución es:



```
emanuel_9809@emanuel-98:~/Dropbox/teoria/parcial_ds/p5$ java MainTuring
Gramatica de balanceo de parentesis
1)Generar cadena de manera manual
2)Generar cadena de manera aleatoria
3)Salir
Ingrese una opcion
1
10101010010B0B010B
```



Su salida es:

```
{ Sx0 SB SB SB }
```