

# REPORTE DE PRÁCTICAS TERCER PARCIAL

ALUMNO: BASTIDA PRADO JAIME ARMANDO

PROFESOR: JUÁREZ MARTÍNEZ GENARO

GRUPO: 2CM5

Diciembre 2017

# Índice

<b>1. Práctica 1: CFG para paréntesis balanceados.</b>	<b>3</b>
1.1. Descripción . . . . .	3
1.2. Ejecución . . . . .	4
1.3. Código . . . . .	7
<b>2. Práctica 2: Máquina de Turing que acepta el lenguaje <math>\{0^n 1^n \mid n \geq 1\}</math>.</b>	<b>12</b>
2.1. Descripción . . . . .	12
2.2. Ejecución . . . . .	13
2.3. Código . . . . .	16

# 1. Práctica 1: CFG para paréntesis balanceados.

## 1.1. Descripción

Este programa puede recibir y analizar una cadena con parentesis por medio de la linea de comandos, un archivo, o una cadena que el mismo genera, al término del análisis el programa muestra el historial de como analizó la cadena mostrando del lado izquierdo el resto de la cadena a leer y del lado derecho la cadena producida por las derivaciones respectivas, esta salida es mostrada en consola y enviada a un archivo también.

Se trata de una gramática no ambigua que genera cadenas de paréntesis balanceados usando las siguientes reglas de derivación:

$$B \rightarrow (RB \mid \epsilon$$

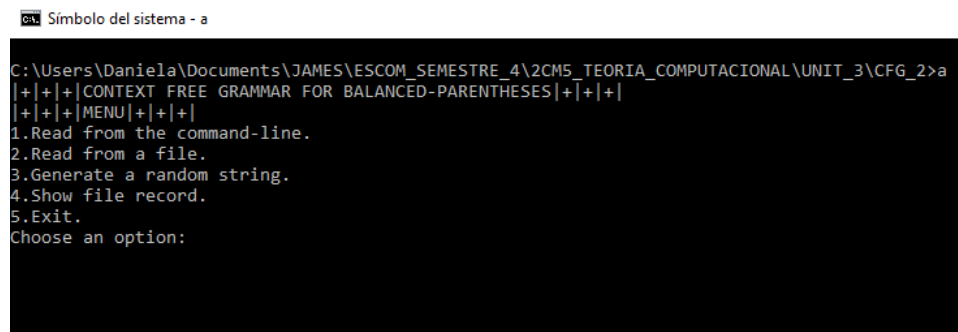
$$R \rightarrow ) \mid ((R$$

Para generar la cadena producida por las reglas de derivación se toma en cuenta lo siguiente:

- Si necesitamos expandir  $B$ , entonces usamos  $B \rightarrow (RB$  si el siguiente símbolo es "(" y usamos  $\epsilon$  si nos encontramos al final de la cadena.
- Si necesitamos expandir  $R$ , usamos  $R \rightarrow )$  si el siguiente símbolo es ")" y  $((R$  si es "(".

## 1.2. Ejecución

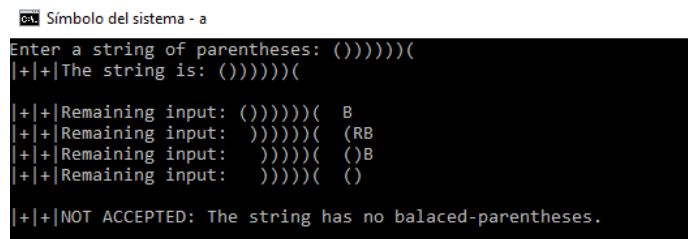
Al iniciar el programa se nos presenta el menú con el siguiente aspecto:



```
Símbolo del sistema - a
C:\Users\Daniela\Documents\JAMES\ESCOM_SEMESTRE_4\2CMS_TEORIA_COMPUTACIONAL\UNIT_3\CFG_2>a
|+|+|CONTEXT FREE GRAMMAR FOR BALANCED-PARENTHESES|+|+|
|+|+|MENU|+|+|
1.Read from the command-line.
2.Read from a file.
3.Generate a random string.
4.Show file record.
5.Exit.
Choose an option:
```

Figura 1: Menú Principal

Si elegimos la opción 1, el programa nos pedirá ingresar una cadena y nos mostrará la salida correspondiente, además de mandar todo a un archivo:



```
Símbolo del sistema - a
Enter a string of parentheses: ())())()
|+|+|The string is: ())())()
|+|+|Remaining input: ())())() B
|+|+|Remaining input:  )())())( (RB
|+|+|Remaining input:  )())())( ()B
|+|+|Remaining input:  )())())( ()
|+|+|NOT ACCEPTED: The string has no balanced-parentheses.
```

Figura 2: Opción 1

Si elegimos la opción 2, el programa nos pedirá ingresar la ubicación junto con el nombre de un archivo a leer y nos mostrará la salida correspondiente, además de mandar todo a un archivo:

```

$ ./Simbolo del sistema - a
Enter name of the file (it may include an adress): read.txt
|+|+The string is: ((()))

|+|+Remaining input: ((()))      B
|+|+Remaining input: ((()))      (RB
|+|+Remaining input: (())        ((RRB
|+|+Remaining input: )           (((RRRB
|+|+Remaining input: )           ((()RRB
|+|+Remaining input: )           ((()RB
|+|+Remaining input: )           ((()B
|+|+Remaining input: (())        ((())

|+|+ACCEPTED: The string has balanced-parentheses.

```

Figura 3: Opción 2

Si elegimos la opción 3, el programa nos pedirá ingresar una longitud para la cadena aleatoria y nos dará la opción de ingresar 0 si queremos que el programa elija:

```

$ ./Simbolo del sistema - a
Enter a length for the random string (0 if you want the program to choose): 10
|+|The random length is: 10
|+|The string is: (()())((

|+|Remaining input: (()())((      B
|+|Remaining input: ()()()      (RB
|+|Remaining input: )()()      (RRB
|+|Remaining input: )()()      (RB
|+|Remaining input: ))()      (RRB
|+|Remaining input: )()      (RB
|+|Remaining input: )(      (B)
|+|Remaining input: )(      (B)

|+|NOT ACCEPTED: The string has no balanced-parentheses.

```

Figura 4: Opción 3

Si elegimos la opción 4, el programa mostrará el archivo con toda la información del proceso que realizó el autómata así como los datos relevantes:

```

Automata_Record: Bloc de notas
Archivo Edición Formato Ver Ayuda
||+|+|The random length is: 10
|+|+|The string is: (()())(((

|+|+|Remaining input: (()())(((      B
|+|+|Remaining input: ()())(((      (RB
|+|+|Remaining input: )())(((      ((RRB
|+|+|Remaining input: ())(((      (())RB
|+|+|Remaining input: ))(((      (())RRB
|+|+|Remaining input: )(((      (())()RB
|+|+|Remaining input: )(((      (())()B
|+|+|Remaining input: )(((      (())()

|+|+|NOT ACCEPTED: The string has no balanced-parentheses.

```

Figura 5: Opción 4

### 1.3. Código

```
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <time.h>

#define RANDOMTOP 10

int main(void)
{
    int option = 0, ch;
    long option_random = 0, random_length = 0,
        derivation_length = 0, i = 0, j = 0, k = 0;
    char *string, *derivation, *aux_derivation;
    char file_name[FILENAME_MAX];
    FILE *read_fp, *write_fp;

    for (;;)
    {
        srand((unsigned) time(NULL));
        printf("++++|CONTEXT_FREE_GRAMMAR_FOR_BALANCED-  

            PARENTHESES|++++|\n");
        printf("++++|MENU|++++|\n");
        printf("1.Read_from_the_command_line.\n");
        printf("2.Read_from_a_file.\n");
        printf("3.Generate_a_random_string.\n");
        printf("4.Show_file_record.\n");
        printf("5.Exit.\n");
        printf("Choose_an_option:_");
        scanf("_%d", &option);

        switch(option)
        {
            case 1: case 2: case 3:
                system("cls");
                getchar(); //Catches the '\n' entered by  

                    the user
                write_fp = fopen("C:/Users/Daniela/  

                    Documents/JAMES/ESCOM/SEMESTRE4/2  

                    CM5.TEORIA.COMPUTACIONAL/UNIT_3/CFG.2/  

                    Automata_Record.txt", "w");
                if(option == 1)
                {
                    string = malloc(100);
                    printf("Enter_a_string_of_  

                        parentheses:_");
                    i = 0;
                    while((ch = getchar()) != '\n')
                        string[i++] = ch;
                    string[i] = '\0';
                    printf("++++|The_string_is:_%\n\n",  

                        , string);
                    fprintf(write_fp, "++++|The_string_  

                        is:_%\n\n", string);
                }
                else if(option == 2)
```

```

{
    printf("Enter_name_of_the_file_(it_may_include_an_address):");
    i = 0;
    while((ch = getchar()) != '\n')
        file_name[i++] = ch;
    file_name[i] = '\0';
    if((read_fp = fopen(file_name, "rb")) == NULL)
    {
        printf("Can't open_%s\n",
            file_name);
        break;
    }
    i = 0;
    while((ch = getc(read_fp)) != EOF)
        i++;
    string = malloc(i);
    fseek(read_fp, 0, SEEK_SET);
    i = 0;
    while((ch = getc(read_fp)) != EOF)
        string[i++] = ch;
    string[i] = '\0';
    printf("|||The_string_is:_%s\n\n",
        string);
    fprintf(write_fp, "|||The_string_is:_%s\n\n", string);
    fclose(read_fp);
}
else if(option == 3)
{
    printf("Enter_a_length_for_the_random_string_(0_if_you_want_the_program_to_choose):");
    scanf("%d", &option_random);
    if(option_random != 0)
        random_length =
            option_random;
    else
        random_length = rand() %
            RANDOMTOP + 1;
    string = malloc(random_length);
    printf("|||The_random_length_is:_%d\n", random_length);
    fprintf(write_fp, "|||The_random_length_is:_%d\n", random_length);
    for(i = 0; i < random_length; i++)
    {
        if(rand() % 2)
            string[i] = '(';
        else
            string[i] = ')';
    }
    string[i] = '\0';
    printf("|||The_string_is:_%s\n\n",
        string);
}

```



```

        fprintf(write_fp, "|||The_string_
                is:_%s\n", string);
    }
    derivation = malloc(2);
    derivation[0] = 'B';
    derivation[1] = '\0';
    derivation_length = 1;
    aux_derivation = malloc(1); //To be able to
                                use realloc
    printf("|||Remaining_input:_%s\t%s\n",
           string, derivation);
    fprintf(write_fp, "|||Remaining_input:_%s
              \t%s\n", string, derivation);
    for(i = 0; string[i] != '\0'; i++)
    {
        if(string[i] == '(' && derivation[i]
           ] == 'B')
        {
            derivation_length += 2;
            derivation = realloc(
                derivation,
                derivation_length + 1);
            derivation[i] = '(';
            derivation[i + 1] = 'R';
            derivation[i + 2] = 'B';
            derivation[i + 3] = '\0';
        }
        else if(string[i] == '(' &&
                derivation[i] == 'R')
        {
            derivation_length += 2;
            derivation = realloc(
                derivation,
                derivation_length + 1);
            aux_derivation = realloc(
                aux_derivation,
                derivation_length + 1);
            //Just in case
            for(j = 0, k = i + 1;
                derivation[k] != '\0';
                j++, k++)
                aux_derivation[j] =
                    derivation[k];
            aux_derivation[j] = '\0';
            derivation[i] = '(';
            derivation[i + 1] = 'R';
            derivation[i + 2] = 'R';
            for(j = i + 3, k = 0;
                aux_derivation[k] != '
                \0'; j++, k++)
                derivation[j] =
                    aux_derivation[
                        k];
            derivation[j] = '\0';
        }
        else if(string[i] == ')') &&
                derivation[i] == 'R')

```

```

        derivation[i] = ')';

    else if(string[i] == ')') &&
        derivation[i] == 'B')
    {
        derivation[i] = '\\0';
        break;
    }
    string[i] = '\\';
    printf("|||Remaining_input:_%s\\t %s\\n", string, derivation);
    fprintf(write_fp, "|||Remaining_input:_%s\\t %s\\n", string,
        derivation);
}
if(derivation[i] == 'B')
{
    derivation[i] = '\\0';
    printf("|||Remaining_input:_%s\\t %s\\n", string, derivation);
    fprintf(write_fp, "|||Remaining_input:_%s\\t %s\\n", string,
        derivation);
    printf("|||ACCEPTED:~The_string_~has_balaced~parentheses.\\n\\n");
    fprintf(write_fp, "|||ACCEPTED:~The_string_has_balaced~parentheses.\\n\\n");
}
else
{
    printf("|||Remaining_input:_%s\\t %s\\n", string, derivation);
    fprintf(write_fp, "|||Remaining_input:_%s\\t %s\\n", string,
        derivation);
    printf("|||NOT_ACCEPTED:~The_string_has_no_balaced~parentheses.\\n\\n");
    fprintf(write_fp, "|||NOT_ACCEPTED:~The_string_has_no_balaced~parentheses.\\n\\n");
}
fclose(write_fp);
free(string);
free(derivation);
free(aux_derivation);
break;
case 4:
    system("C:/Users/Daniela/Documents/JAMES/ESCOM_SEMESTRE_4/2_CM5_TEORIA_COMPUTACIONAL/UNIT_3_CFG.2/Automata_Record.txt");
break;
case 5:
    system("cls");
    exit(EXIT_SUCCESS);

```

```
        break;
    default:
        system("cls");
        printf("Choose_a_correct_option.\n"
        );
        Sleep(2500);
        break;
    }
}

return 0;
}
```

## 2. Práctica 2: Máquina de Turing que acepta el lenguaje $\{0^n 1^n \mid n \geq 1\}$ .

### 2.1. Descripción

Este programa puede recibir y analizar una cadena de 0's y 1's por medio de la línea de comandos, un archivo, o una cadena que el mismo genera, el programa imprime cada movimiento del cabezal en la máquina de Turing, esta salida es mostrada en consola y enviada a un archivo también.

La especificación formal de la máquina es la siguiente:

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

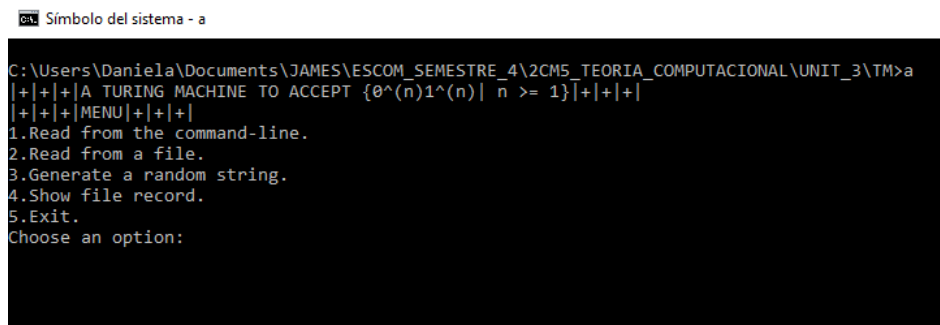
donde  $\delta$  es dada en la siguiente tabla:

<i>State</i>	0	1	<i>X</i>	<i>Y</i>	<i>B</i>
$q_0$	$(q_1, X, R)$	—	—	$(q_3, Y, R)$	—
$q_1$	$(q_1, 0, R)$	$(q_2, Y, L)$	—	$(q_1, Y, R)$	—
$q_2$	$(q_2, 0, L)$	—	$(q_0, X, R)$	$(q_2, Y, L)$	—
$q_3$	—	—	—	$(q_3, Y, R)$	$(q_4, B, R)$
$q_4$	—	—	—	—	—

Cuadro 1: Turing Machine

## 2.2. Ejecución

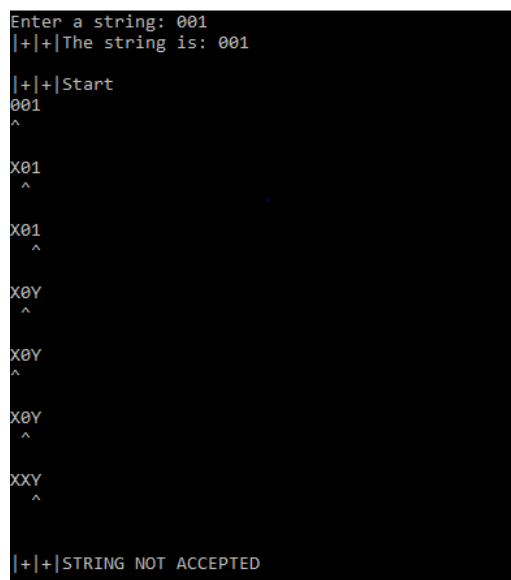
Al iniciar el programa se nos presenta el menú con el siguiente aspecto:



```
cmd Símbolo del sistema - a
C:\Users\Daniela\Documents\JAMES\ESCOM_SEMESTRE_4\2CM5_TEORIA_COMPUTACIONAL\UNIT_3\TM>a
|+|+|A TURING MACHINE TO ACCEPT {0^(n)1^(n)| n >= 1}|+|+|
|+|+|MENU|+|+|
1.Read from the command-line.
2.Read from a file.
3.Generate a random string.
4.Show file record.
5.Exit.
Choose an option:
```

Figura 6: Menú Principal

Si elegimos la opción 1, el programa nos pedirá ingresar una cadena y nos mostrará la salida correspondiente, además de mandar todo a un archivo:



```
Enter a string: 001
|+|+|The string is: 001

|+|+|Start
001
^
X01
^
X01
^
X0Y
^
X0Y
^
X0Y
^
XXY
^

|+|+|STRING NOT ACCEPTED
```

Figura 7: Opción 1

Si elegimos la opción 2, el programa nos pedirá ingresar la ubicación junto con el nombre de un archivo a leer y nos mostrará la salida correspondiente, además de mandar todo a un archivo:

```
Enter name of the file (it may include an adress): r.txt
|+|The string is: 000111

|+|Start
000111
^
X00111
^
X00111
^
X00111
^
X00V11
^
X00V11
^
X00V11
^
X00V11
```

Figura 8: Opción 2

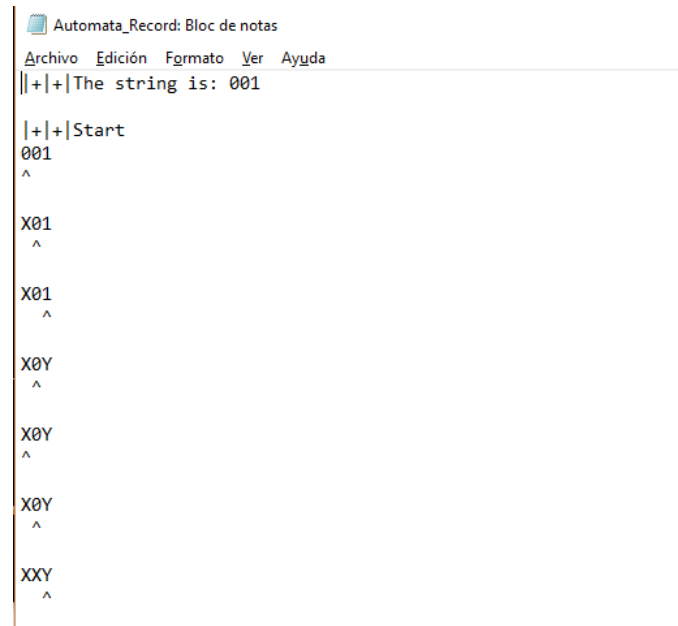
Si elegimos la opción 3, el programa nos pedirá ingresar una longitud para la cadena aleatoria y nos dará la opción de ingresar 0 si queremos que el programa elija:

```
Simbolo del sistema - a
Enter a length for the random string (0 if you want the program to choose): 2
|+|The random length is: 2
|+|The string is: 11

|+|Start
11
^
|+|STRING NOT ACCEPTED
```

Figura 9: Opción 3

Si elegimos la opción 4, el programa mostrará el archivo con toda la información del proceso que realizó el autómata así como los datos relevantes:



```
Automata_Record: Bloc de notas
Archivo Edición Formato Ver Ayuda
|+|+|The string is: 001
|+|+|Start
001
^
X01
^
X01
^
X0Y
^
X0Y
^
X0Y
^
XXY
^
```

Figura 10: Opción 4

## 2.3. Código

```
#include <stdio.h>
#include <windows.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>

#define RANDOMTOP 10
#define Q0 0
#define Q1 1
#define Q2 2
#define Q3 3
#define Q4 4

int main(void)
{
    int option, ch, state;
    bool change;
    long i = 0, option_random = 0, random_length = 0;
    char *string, *machine;
    char file_name[FILENAME_MAX];
    FILE *write_fp, *read_fp;

    for(;;)
    {
        srand((unsigned) time(NULL));
        printf("++++|A_TURING_MACHINE_TO_ACCEPT_{0^(n)1^(n)|_n>=1}|++++|\n");
        printf("++++|MENU|++++|\n");
        printf("1.Read from the command-line.\n");
        printf("2.Read from a file.\n");
        printf("3.Generate a random string.\n");
        printf("4.Show file record.\n");
        printf("5.Exit.\n");
        printf("Choose an option:_");
        scanf("%d", &option);

        switch(option)
        {
            case 1: case 2: case 3:
                system("cls");
                getchar(); //Catches the '\n' entered by the user
                write_fp = fopen("C:/Users/Daniela/Documents/JAMES/ESCOM/SEMESTRE4/2CM5/TEORIA/COMPUTACIONAL/UNIT_3/TM/Automata_Record.txt", "w");
                if(option == 1)
                {
                    string = malloc(100);
                    machine = malloc(100);
                    printf("Enter a string:_");
                    i = 0;
                    while((ch = getchar()) != '\n')
```



```

        {
            machine[i] = '_';
            string[i++] = ch;
        }
        string[i] = '\0';
        machine[i] = '\0';
        printf("+++The_string_is:
        _%s\n\n", string);
        fprintf(write_fp, "+++The
        _string_is:_%s\n\n",
            string);
    }
    else if(option == 2)
    {
        fflush(stdin);
        printf("Enter_name_of_the_
            file_(it_may_include_an
            _address):_");
        i = 0;
        while((ch = getchar()) != '
            \n')
            file_name[i++] = ch
                ;
        file_name[i] = '\0';
        if((read_fp = fopen(
            file_name, "rb")) ==
            NULL)
        {
            printf("Can't_open_
                _%s\n",
                file_name);
            break;
        }
        i = 0;
        while((ch = getc(read_fp))
            != EOF)
            i++;
        string = malloc(i);
        machine = malloc(i);
        fseek(read_fp, 0, SEEK_SET)
            ;
        i = 0;
        while((ch = getc(read_fp))
            != EOF)
        {
            machine[i] = '_';
            string[i++] = ch;
        }
        string[i] = '\0';
        machine[i] = '\0';
        printf("+++The_string_is:
        _%s\n\n", string);
        fprintf(write_fp, "+++The
        _string_is:_%s\n\n",
            string);
        fclose(read_fp);
    }
}

```

```

else if(option == 3)
{
    printf("Enter a length for the random string (0 if you want the program to choose): ");
    scanf("%d", &option_random);
    if(option_random != 0)
        random_length = option_random;
    else
        random_length = rand() % RANDOMTOP + 1;
    string = malloc(random_length);
    machine = malloc(random_length);
    printf("+++The random length is: %d\n", random_length);
    fprintf(write_fp, "+++The random length is: %d\n", random_length);
    for(i = 0; i < random_length; i++)
    {
        if(rand() % 2)
            string[i] = '1';
        else
            string[i] = '0';
        machine[i] = '_';
    }
    string[i] = '\0';
    machine[i] = '\0';
    printf("+++The string is: %s\n\n", string);
    fprintf(write_fp, "+++The string is: %s\n\n", string);
}
printf("+++Start\n");
fprintf(write_fp, "+++Start\n");
printf("%s\n", string);
fprintf(write_fp, "%s\n", string);
machine[0] = '^';
printf("%s\n\n", machine);
fprintf(write_fp, "%s\n\n", machine);
machine[0] = '_';
//AUTOMATA
state = Q0;
for(i = 0; string[i] != '\0';)
{

```

```

//HANDLES POSIBILITIES IN
Q0
if(state == Q0 && string[i]
    == '0')
{
    state = Q1;
    string[i] = 'X';
    i++;
}
else if(state == Q0 &&
    string[i] == 'Y')
{
    state = Q3;
    i++;
}

//HANDLES POSIBILITIES IN
Q1
else if(state == Q1 &&
    string[i] == '0')
{
    i++;
}
else if(state == Q1 &&
    string[i] == '1')
{
    state = Q2;
    string[i] = 'Y';
    i--;
}
else if(state == Q1 &&
    string[i] == 'Y')
{
    i++;
}

//HANDLES POSIBILITIES IN
Q2
else if(state == Q2 &&
    string[i] == '0')
{
    i--;
}
else if(state == Q2 &&
    string[i] == 'X')
{
    state = Q0;
    i++;
}
else if(state == Q2 &&
    string[i] == 'Y')
{
    i--;
}

//HANDLES POSIBILITIES IN
Q3

```

```

else if(state == Q3 &&
        string[i] == 'Y')
{
    i++;
}
if(state == Q3 && string[i]
    == '\0')
{
    state = Q4;
}

//HANDLES EXCEPTIONS
else if(state == Q0 &&
        string[i] == '1')
    break;
else if(state == Q3 &&
        string[i] == '1')
{
    i++;
}
else if(state == Q3 &&
        string[i] == '0')
    i++;

if(state == Q4)
{
    printf("\n|++|
        STRING_ACCEPTED
        \n\n");
    fprintf(write_fp, "
        \n|++|STRING_
        ACCEPTED\n\n");
    break;
}

if(string[i] != '\0')
{
    printf("%s\n",
        string);
    fprintf(write_fp, "
        %s\n", string);
    machine[i] = '^';
    printf("%s\n\n",
        machine);
    fprintf(write_fp, "
        %s\n\n",
        machine);
    machine[i] = '_';
}
}
if(state != Q4)
{
    printf("\n|++|STRING_NOT_
        ACCEPTED\n\n");
    fprintf(write_fp, "\n|++|
        STRING_NOT_ACCEPTED\n\n
        ");
}

```

```

        }
        fclose(write_fp);
    break;
    case 4:
        system("C:/Users/Daniela/Documents/
                JAMES/ESCOM_SEMESTRE4/2
                CM5_TEORIA_COMPUTACIONAL/UNIT_3
                /TM/Automata_Record.txt");
    break;
    case 5:
        system("cls");
        exit(EXIT_SUCCESS);
    break;
    default:
        system("cls");
        printf("Choose a correct option.\n"
              );
        Sleep(2500);
    break;
}
}
return 0;
}

```