REPORTE DE PRÁCTICAS PRIMER PARCIAL

ALUMNO: BASTIDA PRADO JAIME ARMANDO

PROFESOR: JUÁREZ MARTÍNEZ GENARO

GRUPO: 2CM5

Septiembre 2017

Índice

1. Práctica 1: Autómata que calcula los conjuntos potencia desde la potencia 0 hasta la potencia 1000*.

1.1. Descripción

Potencias de un alfabeto

Si Σ es un alfabeto, podemos expresar el conjunto de todas las cadenas de una determinada longitud de dicho alfabeto utilizando una notación exponencial. Definimos Σ^k para que sea el conjunto de las cadenas de longitud k, tales que cada uno de los símbolos de las mismas pertenece a Σ.

Observe que $\Sigma^0 = \{\epsilon\}$, independientemente de cuál sea el alfabeto Σ. Es decir, ϵ es la única cadena cuya longitud es 0.

Si $\Sigma = \{0, 1\}$, entonces:
$\Sigma^1 = \{0, 1\}$,
$\Sigma^2 = \{00, 01, 10, 11\}$,
$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$, etc.

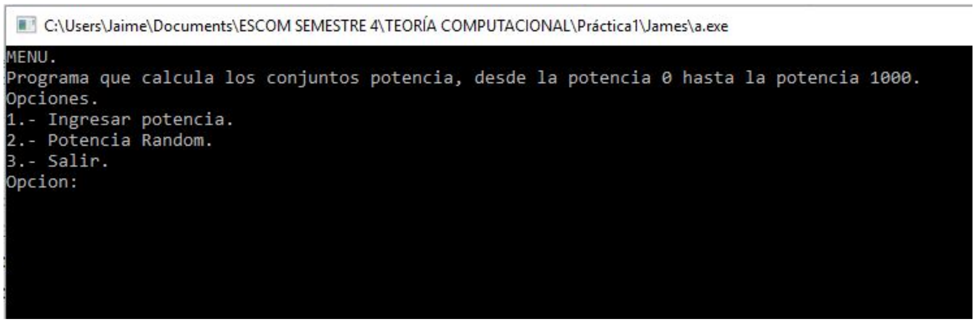
Por convenio, el conjunto de todas las cadenas de un alfabeto Σ se designa mediante Σ^*. Por ejemplo, $\{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, ...\}$. Expresado de otra forma,

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots$$

Este programa calcula Σ^k donde k es un valor ingresado por el usuario, o bien un valor aleatorio generado por el mismo programa, además se imprime cada uno de los k-ésimos alfabetos que el programa calcula y se envía a un archivo de texto. El usuario además puede elegir los símbolos que contendrá el alfabeto no solo 0 o 1.

1.2. Ejecución

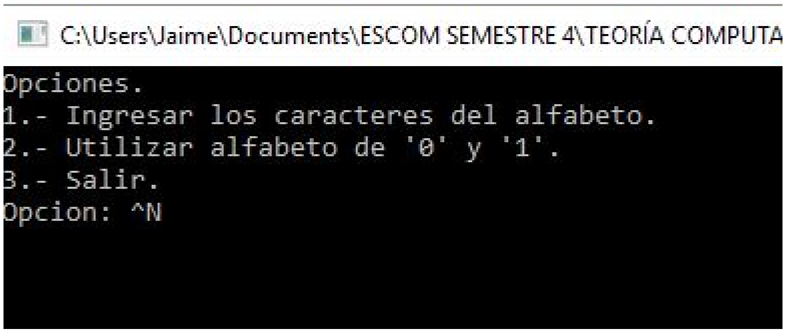
Al iniciar el programa se nos presenta el menú con el siguiente aspecto:


C:\Users\Jaime\Documents\ESCOM SEMESTRE 4\TEORÍA COMPUTACIONAL\Práctica1\James\a.exe

```
MENU.
Programa que calcula los conjuntos potencia, desde la potencia 0 hasta la potencia 1000.
Opciones.
1.- Ingresar potencia.
2.- Potencia Random.
3.- Salir.
Opcion:
```

Figura 1: MenuPrincipal

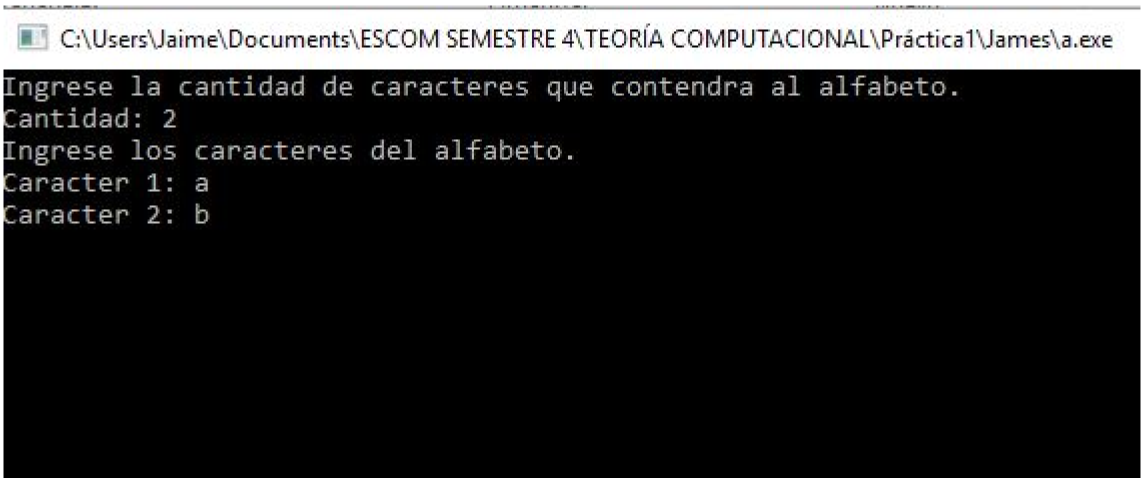
Eligiendo la opción 1, podemos ingresar cualquier potencia, por ejemplo 5, entonces entraremos al siguiente menú:



```
Opciones.
1.- Ingresar los caracteres del alfabeto.
2.- Utilizar alfabeto de '0' y '1'.
3.- Salir.
Opcion: ^N
```

Figura 2: Menu2Opción1

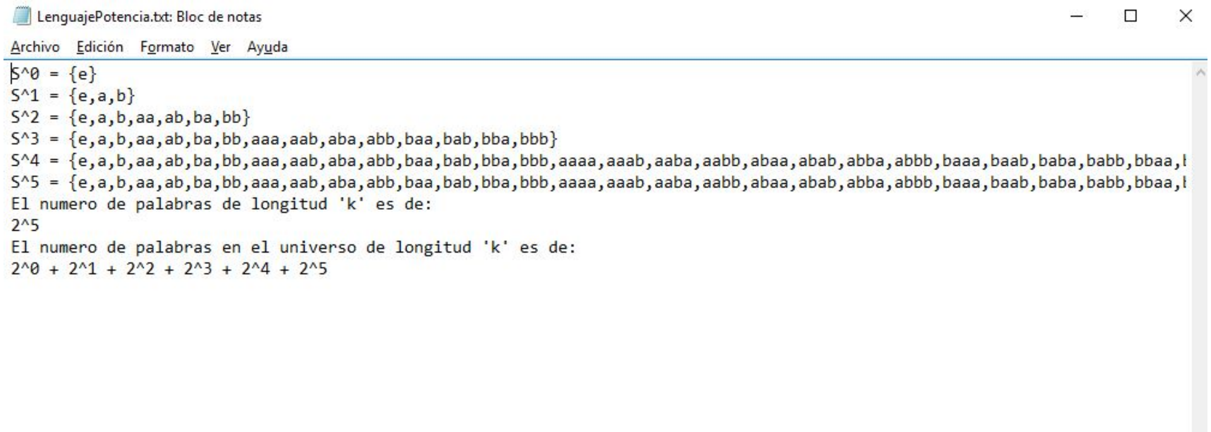
Desde este menú podemos elegir entre tres opciones, la primera utilizando cualesquiera de las letras del alfabeto, ejemplo:


C:\Users\Jaime\Documents\ESCOM SEMESTRE 4\TEORÍA COMPUTACIONAL\Práctica1\James\a.exe

```
Ingrese la cantidad de caracteres que contendra al alfabeto.
Cantidad: 2
Ingrese los caracteres del alfabeto.
Caracter 1: a
Caracter 2: b
```

Figura 3: Menu3

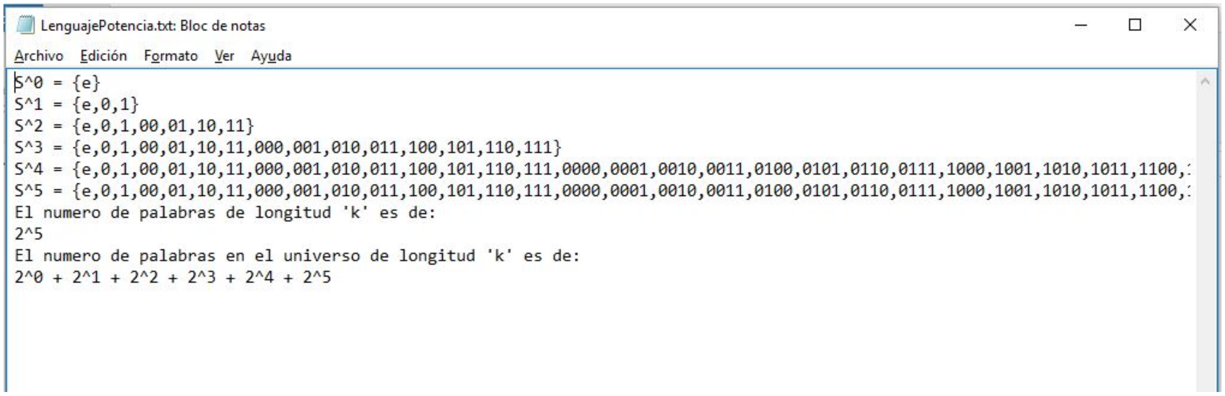
Obtendremos entonces un bloc de notas con la información deseada:



```
S^0 = {e}
S^1 = {e,a,b}
S^2 = {e,a,b,aa,ab,ba,bb}
S^3 = {e,a,b,aa,ab,ba,bb,aaa,aab,aba,abb,baa,bab,bba,bbb}
S^4 = {e,a,b,aa,ab,ba,bb,aaa,aab,aba,abb,baa,bab,bba,bbb,aaaa,aaab,aaba,aabb,abaa,abab,abba,abbb,baaa,baab,baba,babb,bbaa,
S^5 = {e,a,b,aa,ab,ba,bb,aaa,aab,aba,abb,baa,bab,bba,bbb,aaaa,aaab,aaba,aabb,abaa,abab,abba,abbb,baaa,baab,baba,babb,bbaa,
El numero de palabras de longitud 'k' es de:
2^5
El numero de palabras en el universo de longitud 'k' es de:
2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5
```

Figura 4: Menu3Bloc

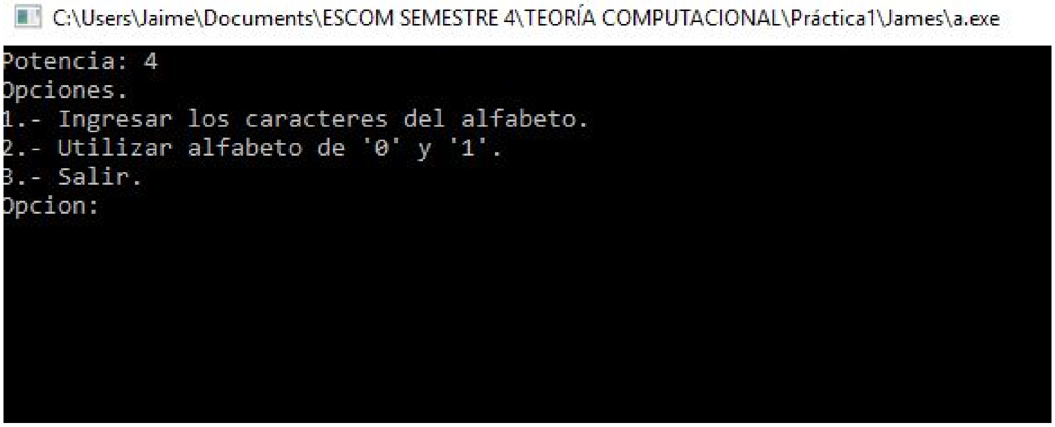
La segunda opción del menú 2 trabajaría el alfabeto con '0'y '1'únicamente:


LenguajePotencia.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
S^0 = {e}
S^1 = {e,0,1}
S^2 = {e,0,1,00,01,10,11}
S^3 = {e,0,1,00,01,10,11,000,001,010,011,100,101,110,111}
S^4 = {e,0,1,00,01,10,11,000,001,010,011,100,101,110,111,0000,0001,0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,
S^5 = {e,0,1,00,01,10,11,000,001,010,011,100,101,110,111,0000,0001,0010,0011,0100,0101,0110,0111,1000,1001,1010,1011,1100,
El numero de palabras de longitud 'k' es de:
2^5
El numero de palabras en el universo de longitud 'k' es de:
2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5
```

Figura 5: Menu2Opción2Bloc

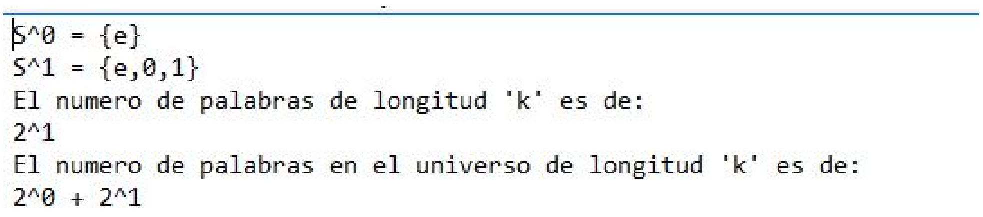
Desde el menú 1 podemos elegir la opción 2 haciendo que la potencia sea random con el mismo número de opciones en el menú 2, y la potencia random indicada en la parte superior:


C:\Users\Jaime\Documents\ESCOM SEMESTRE 4\TEORÍA COMPUTACIONAL\Práctica1\James\a.exe

```
Potencia: 4
Opciones.
1.- Ingresar los caracteres del alfabeto.
2.- Utilizar alfabeto de '0' y '1'.
3.- Salir.
Opcion:
```

Figura 6: Menu2Opción2Bloc

Aquí se muestra un resultado con la potencia random:



```
S^0 = {e}
S^1 = {e,0,1}
El numero de palabras de longitud 'k' es de:
2^1
El numero de palabras en el universo de longitud 'k' es de:
2^0 + 2^1
```

Figura 7: RandomBloc

1.3. Código

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

void print(int power, char alphabet[], int length_alphabet)
{
        int w, x, y, z, v;
        FILE *fp;
        if((fp = fopen(”LenguajePotencia.txt”, ”w”)) != NULL)
        {
                for(v = 0; v <= power; v++) //CONTROLA EL NUMERO DE
                     ALFABETOS SEG N LA POTENCIA
                {
                printf(”S^%d␣=␣{”, v);
                fprintf(fp,”S^%d␣=␣{”, v);
                for(w = 0; w <= v; w++)              //CONTROLA EL
                    N MERO DE CADENAS DE LONGITUD V POR ALFABETO
                {
                if(w == 0){
                printf(”e”);
                fprintf(fp,”e”);
                }
                for(x = 0; x < pow(length_alphabet, w); x++)      //
                    CONTROLA EL N MERO DE CADENAS SEGUN LA
                    POTENCIA
                {
                char *array;
                int i, b;
                array = (char *)malloc(w * sizeof(char));
                b = x;
                for(i = 0; i < w ; i++)                          //
                    CONTROLA EL N MERO DE S MBOLOS POR CADENA Y
                    LOS ACOMODA
                {
                if(b == 1)
                        array[i] = alphabet[b];
                else
                        array[i] = alphabet[b % length_alphabet];
                b = b / length_alphabet;
                }

                for(i = (w - 1); i >= 0; i--)            //IMPRIME
                    S MBOLO POR S MBOLO LA CADENA YA ACOMODADA
                {
                printf(”%c”, array[i]);
                fprintf(fp,”%c”, array[i]);
                }
                if(x == (pow(length_alphabet, w) - 1));          //
                    SI ENTRA AL ELSE SIGNIFICA QUE HAY CADENAS POR
                    IMPRIMIR DE LA MISMA POTENCIA
                else
                {
                printf(”,”);
                fprintf(fp,”,”);
                }
                }
                if(v == w);
                else                                        //SI
                    ENTRA AL ELSE SIGNIFICA QUE FALTAN CADENAS POR
                    IMPRIMIR PERO DE LA SIG POTENCIA
                {
                printf(",");
                fprintf(fp,",");
                }
                }
                printf("}\n");
                fprintf(fp,"}\n");
                }
        fclose(fp);
        }
}

void Set(int power, int length_alphabet)
{
        FILE *fp;

        if( (fp = fopen("LenguajePotencia.txt", "a")) != NULL)
        {
                printf("El␣numero␣de␣palabras␣de␣longitud␣'k'␣es␣de
                    :\n");
                fprintf(fp,"El␣numero␣de␣palabras␣de␣longitud␣'k'␣
                    es␣de:\n");
                printf("%d^%d\n", length_alphabet, power);
                fprintf(fp,"%d^%d\n", length_alphabet, power);
        }
}

void Universe(int power, int length_alphabet)
{
        FILE *fp;

        if( (fp = fopen("LenguajePotencia.txt","a")) != NULL)
        {
                printf("El␣numero␣de␣palabras␣en␣el␣universo␣de␣
                    longitud␣'k'␣es␣de:\n");
                fprintf(fp,"El␣numero␣de␣palabras␣en␣el␣universo␣de
                    ␣longitud␣'k'␣es␣de:\n");
                int i;
                for(i = 0; i <= power; i++)
                {
                if(i == power)
                {
                printf("%d^%d", length_alphabet, i);
                fprintf(fp,"%d^%d", length_alphabet, i);
                printf("\n");
                fprintf(fp,"\n");
                }
                else
                {
                printf("%d^%d␣+␣", length_alphabet, i);
                fprintf(fp,"%d^%d␣+␣", length_alphabet, i);
                }
                }
        }
}

int main(void)
{
        int option, power, i = 0;
        option = 1;
        while(option > 0 && option < 3)
        {
                printf("MENU.\n");
                printf("Programa que calcula los conjuntos potencia
                    , desde la potencia 0 hasta la potencia 1000.\n
                    ");
                printf("Opciones.\n");
                printf("1.- Ingresar potencia.\n");
                printf("2.- Potencia Random.\n");
                printf("3.- Salir.\n");

                printf("Opcion: ");
                scanf("%d", &option);

                system("cls");

                switch(option)
                {
                case 1:
                printf("Ingrese la potencia para el conjunto 
                    potencia.\n");
                printf("Potencia: ");
                scanf("%d", &power);

                int option2 = 1;
                while(option2 >= 1 && option2 <= 3)
                {
                system("cls");
                printf("Opciones.\n");
                printf("1.- Ingresar los caracteres del alfabeto.\n
                    ");
                printf("2.- Utilizar alfabeto de '0' y '1'.\n");
                printf("3.- Salir.\n");

                printf("Opcion: ");
                scanf("%d", &option2);

                if(option2 == 3)
                {
                system("cls");
                }

                switch(option2)
                {
                case 1:
                {
                system("cls");
                int n_characters;
                printf("Ingrese␣la␣cantidad␣de␣caracteres␣que␣
                    contendra␣al␣alfabeto.\n");
                printf("Cantidad:␣");
                scanf("%d", &n_characters);

                char alphabet[n_characters];

                printf("Ingrese␣los␣caracteres␣del␣alfabeto.\n");
                for(i = 0; i < n_characters; i++){
                printf("Caracter␣%d:␣", i + 1);
                scanf("␣%c", &alphabet[i]);
                }

                system("cls");
                printf("Los␣caracteres␣del␣alfabeto␣son:\n");
                i = 0;
                int length_alphabet = sizeof(alphabet);
                printf("S␣=␣{");
                while(i < length_alphabet){
                printf("%c,", alphabet[i]);
                i++;
                }
                printf("\b}");
                printf("\n");
                printf("Conjuntos␣potencia.\n");
                print(power, alphabet, length_alphabet);
                Set(power, length_alphabet);
                Universe(power, length_alphabet);
                return 0;
                }
                break;
                case 2:
                system("cls");
                char alphabet2[2] = {'0','1'};
                printf("Los␣caracteres␣del␣alfabeto␣son:\n");
                int j = 0;
                int length_alphabet2 = sizeof(alphabet2);
                printf("S␣=␣{");
                while(j < length_alphabet2)
                {
                        printf("%c,", alphabet2[j]);
                        j++;
                }
                printf("\b}");
                printf("\n");
                printf("Conjuntos␣potencia.\n");
                print(power, alphabet2 ,length_alphabet2);
                Set(power, length_alphabet2);
                Universe(power, length_alphabet2);
                return 0;
                system("cls");
                break;
                case 3:
                break;
                default:
                printf("Error.\n");
                }
```

```
}
break;
case 2:
srand((unsigned) time(NULL));
system("cls");
power = rand() % 10;
printf("Potencia:␣%d\n", power);

option2 = 1;
while(option2 >= 1 && option2 < 3)
{
printf("Opciones.\n");
printf("1.-␣Ingresar␣los␣caracteres␣del␣alfabeto.\n
    ");
printf("2.-␣Utilizar␣alfabeto␣de␣'0'␣y␣'1'.\n");
printf("3.-␣Salir.\n");

printf("Opcion:␣");
scanf("%d", &option2);

if(option2 == 3)
{
system("cls");
}

switch(option2)
{
case 1:
{
system("cls");
int n_characters;
printf("Ingrese␣la␣cantidad␣de␣caracteres␣que␣
    contendra␣al␣alfabeto.\n");
printf("Cantidad:␣");
scanf("%d", &n_characters);

char alphabet3[n_characters];

printf("Ingrese␣los␣caracteres␣del␣alfabeto.\n");
int k;
for(k = 0; k < n_characters; k++)
{
printf("Caracter␣%d:␣", k + 1);
scanf("␣%c",&alphabet3[k]);
}
system("cls");
printf("Los␣caracteres␣del␣alfabeto␣son:\n");
k = 0;
int length_alphabet3 = sizeof(alphabet3);
printf("S␣=␣{");
while(k < length_alphabet3)
{
printf("%c,", alphabet3[k]);
k++;
}
printf("\b}");
printf("\n");
```

```
                printf("Conjuntos␣potencia.\n");
                print(power, alphabet3, length_alphabet3);
                Set(power, length_alphabet3);
                Universe(power, length_alphabet3);
                return 0;
                system("cls");
                }
                break;
                case 2:
                system("cls");
                char alphabet4[2]={'0','1'};
                printf("Los␣caracteres␣del␣alfabeto␣son:\n");
                int o = 0;
                int length_alphabet4 = sizeof(alphabet4);
                printf("S␣=␣{");
                while(o < length_alphabet4)
                {
                        printf("%c,", alphabet4[o]);
                        o++;
                }
                printf("\b}");
                printf("\n");
                printf("Conjuntos␣potencia.\n");
                print(power, alphabet4, length_alphabet4);
                Set(power, length_alphabet4);
                Universe(power, length_alphabet4);
                return 0;
                system("cls");
                break;
                case 3:
                break;
                default:
                printf("Error.\n");
                }
                }
                break;
                case 3:
                break;
                default:
                printf("Error.\n");
                }
                system("cls");
        }
}
```

2. Práctica 3: Autómata que reconoce cadenas que acaban en ïng”.

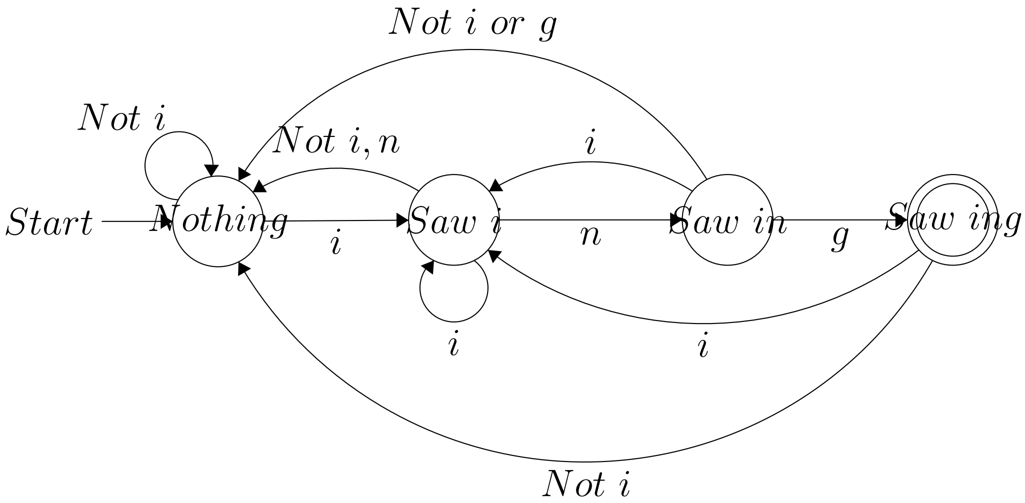
2.1. Descripción

Esta programa resuelve el problema de reconocer todas las cadenas en un texto que terminan en ïng¿a sea ingresado por el usuario desde consola o leído desde un archivo de texto, el programa imprime el historial de como trabaja el autómata palabra por palabra en consola y además en un archivo de texto indicando cuando encuentra una palabra aceptada por el lenguaje,

$$L = \{w|\, w\, es\, una\, palabra\, que\, acaba\, en\, ”ing”\}$$

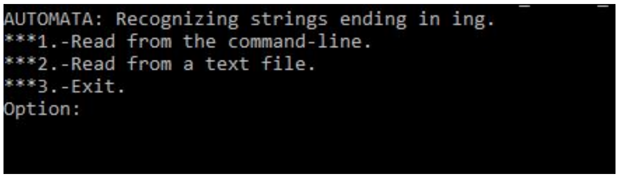
Al final del historial se muestra cuantas palabras fueron reconocidas en el texto.

A continuación el grafo del autómata:



2.2. Ejecución

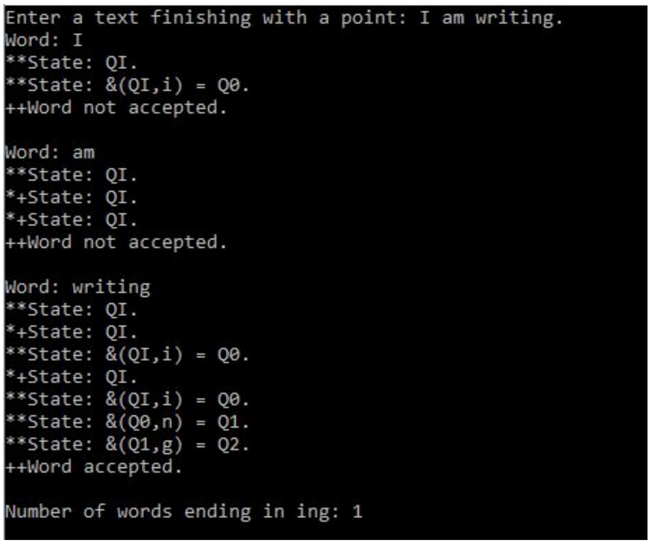
Al ejecutar el programa se nos presenta un menú con el siguiente aspecto:



```
AUTOMATA: Recognizing strings ending in ing.
***1.-Read from the command-line.
***2.-Read from a text file.
***3.-Exit.
Option:
```

Figura 8: MenuPrincipal

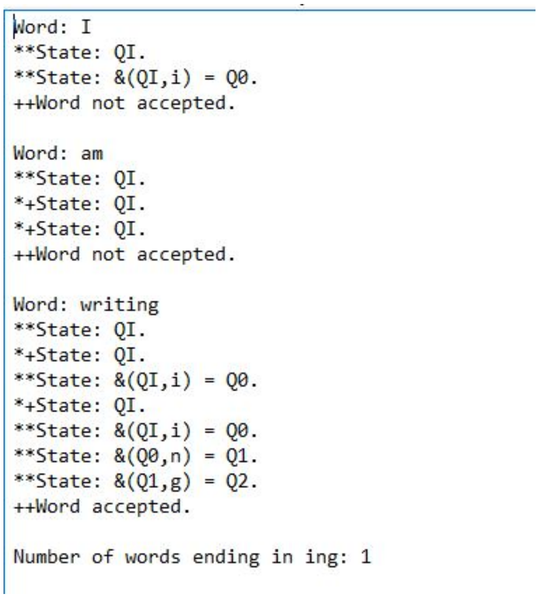
Si elegimos la opción 1, el programa pedirá que se ingrese un texto en consola que acabe en ”.”, en el ejemplo ingresamos: Ï am writting.”



```
Enter a text finishing with a point: I am writing.
Word: I
**State: QI.
**State: &(QI,i) = Q0.
++Word not accepted.

Word: am
**State: QI.
*+State: QI.
*+State: QI.
++Word not accepted.

Word: writing
**State: QI.
*+State: QI.
**State: &(QI,i) = Q0.
*+State: QI.
**State: &(QI,i) = Q0.
**State: &(Q0,n) = Q1.
**State: &(Q1,g) = Q2.
++Word accepted.

Number of words ending in ing: 1
```

Figura 9: Salida a consola de la opción 1.

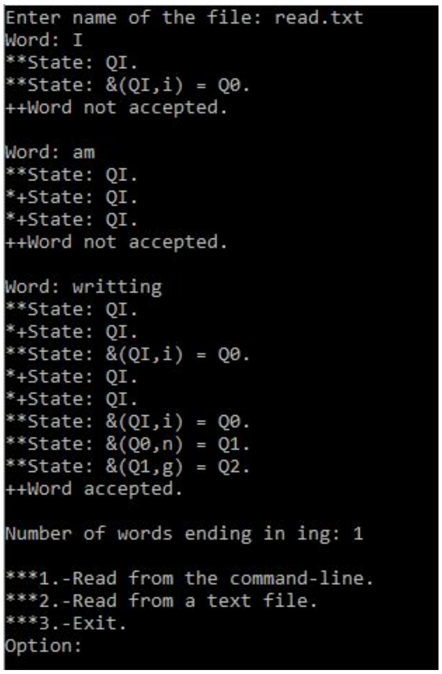
Y mandará el historial también a un archivo de texto:



```
Word: I
**State: QI.
**State: &(QI,i) = Q0.
++Word not accepted.

Word: am
**State: QI.
*+State: QI.
*+State: QI.
++Word not accepted.

Word: writing
**State: QI.
*+State: QI.
**State: &(QI,i) = Q0.
*+State: QI.
**State: &(QI,i) = Q0.
**State: &(Q0,n) = Q1.
**State: &(Q1,g) = Q2.
++Word accepted.

Number of words ending in ing: 1
```

Figura 10: Salida a archivo de texto de la opción 1.

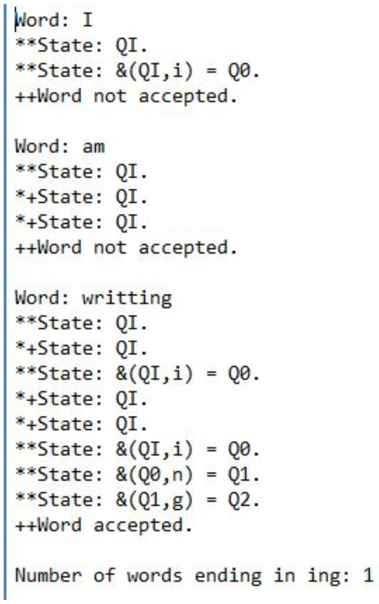
Si elegimos la opción 2, el programa pedirá que se ingrese el nombre del archivo de texto, en este caso -ead.txt", y nos mostrará el historial en consola además de mandarlo a un archivo de texto:



```
Enter name of the file: read.txt
Word: I
**State: QI.
**State: &(QI,i) = Q0.
++Word not accepted.

Word: am
**State: QI.
*+State: QI.
*+State: QI.
++Word not accepted.

Word: writting
**State: QI.
*+State: QI.
**State: &(QI,i) = Q0.
*+State: QI.
*+State: QI.
**State: &(QI,i) = Q0.
**State: &(Q0,n) = Q1.
**State: &(Q1,g) = Q2.
++Word accepted.

Number of words ending in ing: 1

***1.-Read from the command-line.
***2.-Read from a text file.
***3.-Exit.
Option:
```

Figura 11: Salida a consola de la opción 2.

Salida de texto:



```
Word: I
**State: QI.
**State: &(QI,i) = Q0.
++Word not accepted.

Word: am
**State: QI.
*+State: QI.
*+State: QI.
++Word not accepted.

Word: writting
**State: QI.
*+State: QI.
**State: &(QI,i) = Q0.
*+State: QI.
*+State: QI.
**State: &(QI,i) = Q0.
**State: &(Q0,n) = Q1.
**State: &(Q1,g) = Q2.
++Word accepted.

Number of words ending in ing: 1
```

Figura 12: Salida a archivo de texto de la opción 2.

2.3. Código

```
#include <stdio.h>
#include <ctype.h>
#include <stdbool.h>
#include <stdlib.h>

#define STR_LENGTH 51

int getString(int n, char string[n],  int first_ch);
int getStringFile(int n, char string[n],  int first_ch, FILE *
    read_fp);
bool automata(char *p, int mode);

int main (void)
{
        char string[STR_LENGTH];
        int first_ch, ch, i = 0, words = 0, count_ing = 0, option,
            mode, last_ch;
        FILE *write_fp, *read_fp;
        char read_file[51] = {'␣'};

        printf("AUTOMATA:␣Recognizing␣strings␣ending␣in␣ing.\n");
        for(;;)
        {
                printf("***1.-Read␣from␣the␣command-line.\n***2.-
                    Read␣from␣a␣text␣file.\n***3.-Exit.\n");
                printf("Option:␣");
                scanf("␣%d", &option);
                system("cls");
                count_ing = 0;
                switch (option)
                {
                case 1:
                words = 0;
                printf("Enter␣a␣text␣finishing␣with␣a␣point:␣");
                for(;;)
                {
                if((first_ch = getchar()) == '.')
                        break;
                last_ch = getString(STR_LENGTH, string, first_ch);
                                        //OBTIENE UNA CADENA YA DEPURADA
                    (SIN OTRA COSA QUE LETRAS)
                words += 1;
                if (words == 1)
                        mode = 'w';
                else
                        mode = 'a';
                if(automata(string, mode))
                        count_ing += 1;
                if (last_ch == '.')
                        break;
                }

                write_fp = fopen("Automata_record.txt", "a");
                printf("Number␣of␣words␣ending␣in␣ing:␣%d\n\n",
                    count_ing);
                fprintf(write_fp, "Number␣of␣words␣ending␣in␣ing:␣%
                    d\n\n", count_ing);
                fclose(write_fp);
                break;

                case 2:
                words = 0;
                fflush(stdin);                          //SE
                    ENCUENTRA AQU  DADO QUE EL BUFFER SE QUEDA CON
                      EL '\n' QUE INGRESAMOS AL ELEGIR OPCI N
                printf("Enter␣name␣of␣the␣file:␣");
                while((ch = getchar()) != '\n')
                {
                read_file[i++] = ch;
                }
                i = 0;
                if ((read_fp = fopen(read_file, "rb")) == NULL)
                {
                fprintf(stderr, "Can't␣open:␣%s\n", read_file);
                break;
                }

                while((first_ch = getc(read_fp)) != EOF)
                {
                getStringFile(STR_LENGTH, string, first_ch, read_fp
                    );
                words += 1;
                if(words == 1)
                        mode = 'w';
                else
                        mode = 'a';
                if(automata(string, mode))
                        count_ing += 1;
                }
                fclose(read_fp);

                write_fp = fopen("Automata_record.txt", "a");
                printf("Number␣of␣words␣ending␣in␣ing:␣%d\n\n",
                    count_ing);
                fprintf(write_fp, "Number␣of␣words␣ending␣in␣ing:␣%
                    d\n", count_ing);
                fclose(write_fp);
                break;

                case 3: exit(EXIT_SUCCESS);
                break;
                }
        }
        return 0;
}


int getString(int n, char string[n], int first_ch)
{
        int ch, i = 0;
        if((first_ch != '␣') && ((first_ch >= 65 && first_ch <= 90)
             || (first_ch >= 97 && first_ch <= 122)))
                string[i++] = first_ch;
        while(((ch = getchar()) != '␣') && ((ch >= 65 && ch <= 90)
            || (ch >= 97 && ch <= 122)) && i < n && ch != '.')
                string[i++] = ch;
        string[i] = '\0';

        return ch;
}

int getStringFile(int n, char string[n], int first_ch, FILE *
    read_fp)
{
        int ch, i = 0;

        if( (first_ch != '␣') && ((first_ch >= 65 && first_ch <=
            90) || (first_ch >= 97 && first_ch <= 122)) )
                string[i++] = first_ch;
        while(((ch = getc(read_fp)) != '␣') && ((ch >= 65 && ch <=
            90) || (ch >= 97 && ch <= 122)) && i < n && ch != EOF)
                string[i++] = ch;
        string[i] = '\0';

        return i;
}

bool automata(char *p, int mode)
{
        int state = -1;
        bool accepted = false;
        FILE *fp;

        if (mode == 'w')
                fp = fopen("Automata_record.txt", "w");
        else
                fp = fopen("Automata_record.txt", "a");
        printf("Word:␣%s\n", p);
        fprintf(fp, "Word:␣%s\n", p);
        if(*p != '\0')
        {
                printf("**State:␣QI.\n");
                fprintf(fp, "**State:␣QI.\n");
        }

        while(*p != '\0')                                        //
            RECORRE LA CADENA
        {
                if((*p == 'i' || *p == 'I') && state == -1)
                                    //SI ENTRA EN ESTE IF, SE
                    ENCONTRAR EN EL ESTADO Q0
                {
                        p++;
                        state = 0;
                        printf("**State:␣&(QI,i)␣=␣Q0.\n");
                        fprintf(fp, "**State:␣&(QI,i)␣=␣Q0.\n");
                }
```

```
if((*p == ’n’ || *p == ’N’) && state == 0)
                        //SI ENTRA EN ESTE IF, SE
    ENCONTRAR EN EL ESTADO Q1
{
        p++;
        state = 1;
        printf(”**State:␣&(Q0,n)␣=␣Q1.\n”);
        fprintf(fp, ”**State:␣&(Q0,n)␣=␣Q1.\n”);
}
if((*p == ’g’ || *p == ’G’) && state == 1)
                        //SI ENTRA EN ESTE ESTADO, SE
    ENCONTRAR EN EL ESTADO Q2 Y FINAL
{
        p++;
        state = 2;
        printf(”**State:␣&(Q1,g)␣=␣Q2.\n”);
        fprintf(fp, ”**State:␣&(Q1,g)␣=␣Q2.\n”);
}

if(state == 2 && *p == ’\0’)
                                    //SI ENTRA EN ESTE IF,
    SIGNIFICA QUE LA PALABRA HA SIDO ACEPTADA
{
        printf(”++Word␣accepted.\n\n”);
        fprintf(fp, ”++Word␣accepted.\n\n”);
        accepted = true;
}

if((toupper(*p) != ’I’) && state == -1  && *p != ’
    \0’)
        p++;

if((toupper(*p) == ’I’) && state == 0 && *p != ’\0’
    )
        state = -1;
if((toupper(*p) != ’I’) && state == 0 && *p != ’\0’
    )
        state = -1;

if((toupper(*p) == ’I’) && state == 1 && *p != ’\0’
    )
        state = 0;
if((toupper(*p) != ’I’) && state == 1 && *p != ’\0’
    )
        state = -1;

if((toupper(*p) == ’I’) && state == 2 && *p != ’\0’
    )
        state = 0;
if((toupper(*p) != ’I’) && state == 2 && *p != ’\0’
    )
        state = -1;

if((state == -1 && toupper(*p) != ’I’) || (state ==
     2 && toupper(*p) != ’I’ && *p != ’\0’))
{
        printf(”*+State:␣QI.\n”);
```

```
            fprintf(fp, "*+State:␣QI.\n");
        }
        if(*p == '\0' && accepted == false)
        {
            printf("++Word␣not␣accepted.\n\n");
            fprintf(fp, "++Word␣not␣accepted.\n\n");
        }
    }
    fclose(fp);
    return accepted;
}
```

3. Práctica 4: Autómata que reconoce cadenas con un número par de 0's y un número par de 1's.

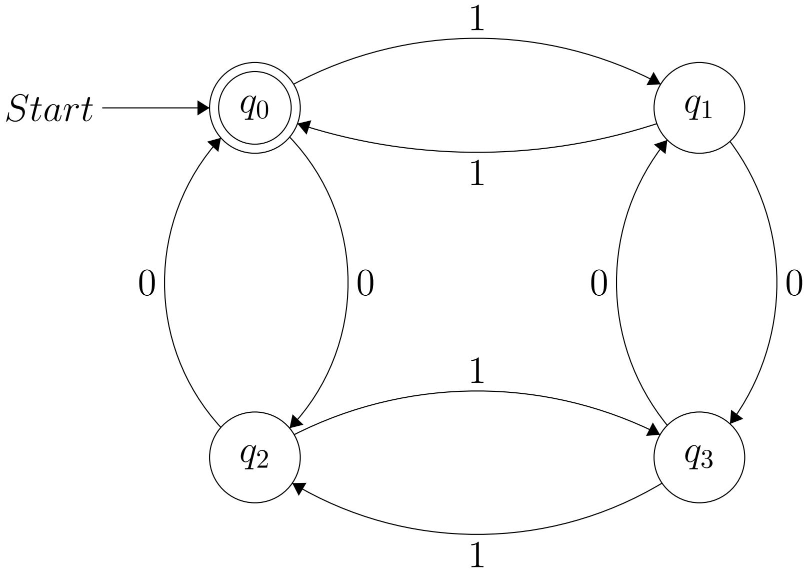
3.1. Descripción

Este programa reconoce el lenguaje,

$$L = \{w \,|\, w\, has\, both\, an\, even\, number\, of\, 0's\, and\, an\, even\, number\, of\, 1's\}$$

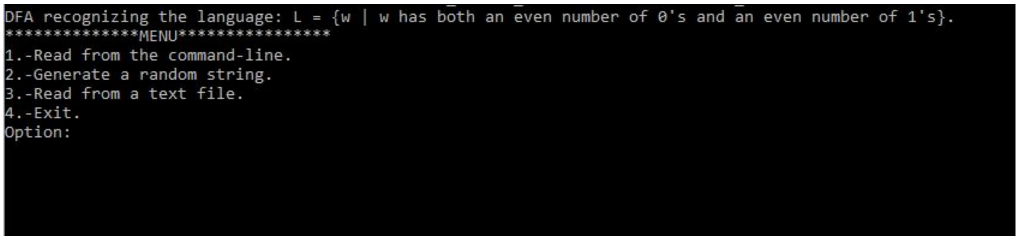
Es decir cadenas con un número par de 0's y un número par de 1's, puede leer la cadena desde consola, generar una aleatoriamente que podrá ser de hasta 10000 carácteres de largo, o leer varias cadenas desde un archivo de texto, al terminar de calcular imprimira en consola y en un archivo de texto el historial de como trabajo el autómata.

A continuación el grafo del autómata:



3.2. Ejecución

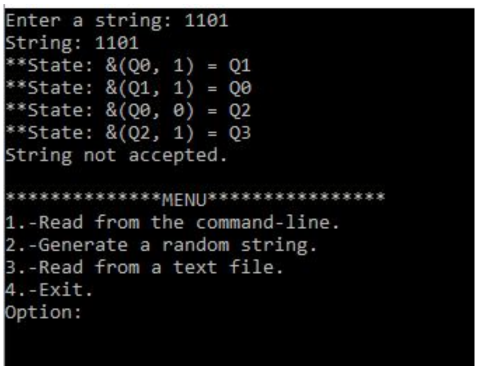
Al iniciar el programa nos encontraremos con el siguiente menú:



```
DFA recognizing the language: L = {w | w has both an even number of 0's and an even number of 1's}.
*************MENU***************
1.-Read from the command-line.
2.-Generate a random string.
3.-Read from a text file.
4.-Exit.
Option:
```

Figura 13: Menú Principal.

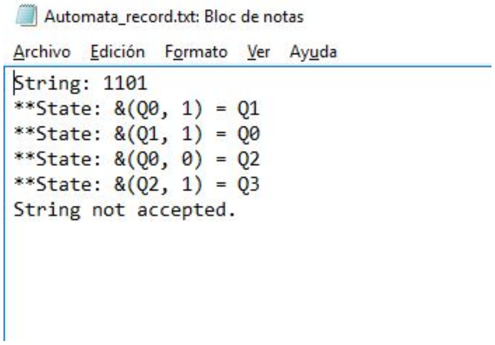
Al seleccionar la opción 1 nos pedirá ingresar una cadena y a continuación nos mostrará el historial en consola y archivo:



```
Enter a string: 1101
String: 1101
**State: &(Q0, 1) = Q1
**State: &(Q1, 1) = Q0
**State: &(Q0, 0) = Q2
**State: &(Q2, 1) = Q3
String not accepted.

*************MENU***************
1.-Read from the command-line.
2.-Generate a random string.
3.-Read from a text file.
4.-Exit.
Option:
```

Figura 14: Salida a consola de la opción 1.

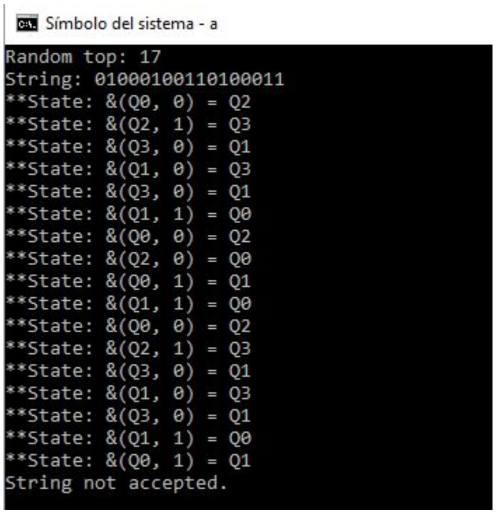
Salida a archivo de texto:


Automata_record.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
String: 1101
**State: &(Q0, 1) = Q1
**State: &(Q1, 1) = Q0
**State: &(Q0, 0) = Q2
**State: &(Q2, 1) = Q3
String not accepted.
```

Figura 15: Salida a archivo de texto de la opción 1.

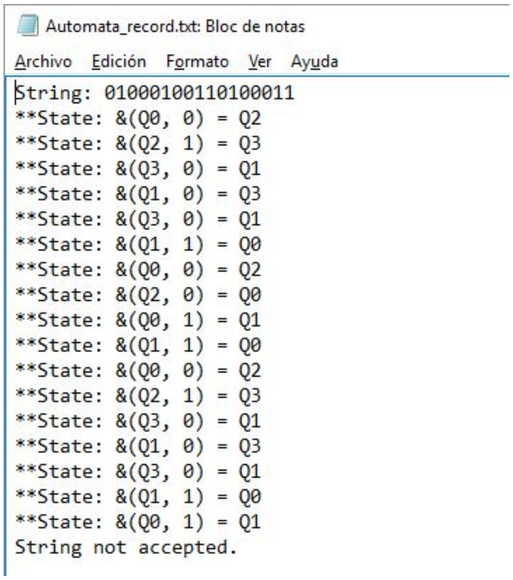
Si seleccionamos la opción 2 generará una tope máximo de carácteres y una cadena aleatoria, junto con el historial a consola y archivo de texto:



```
Random top: 17
String: 01000100110100011
**State: &(Q0, 0) = Q2
**State: &(Q2, 1) = Q3
**State: &(Q3, 0) = Q1
**State: &(Q1, 0) = Q3
**State: &(Q3, 0) = Q1
**State: &(Q1, 1) = Q0
**State: &(Q0, 0) = Q2
**State: &(Q2, 0) = Q0
**State: &(Q0, 1) = Q1
**State: &(Q1, 1) = Q0
**State: &(Q0, 0) = Q2
**State: &(Q2, 1) = Q3
**State: &(Q3, 0) = Q1
**State: &(Q1, 0) = Q3
**State: &(Q3, 0) = Q1
**State: &(Q1, 1) = Q0
**State: &(Q0, 1) = Q1
String not accepted.
```

Figura 16: Salida a consola de la opción 2.

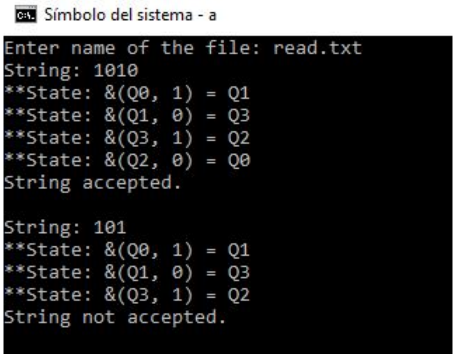
Salida a archivo de texto:


Automata_record.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
String: 01000100110100011
**State: &(Q0, 0) = Q2
**State: &(Q2, 1) = Q3
**State: &(Q3, 0) = Q1
**State: &(Q1, 0) = Q3
**State: &(Q3, 0) = Q1
**State: &(Q1, 1) = Q0
**State: &(Q0, 0) = Q2
**State: &(Q2, 0) = Q0
**State: &(Q0, 1) = Q1
**State: &(Q1, 1) = Q0
**State: &(Q0, 0) = Q2
**State: &(Q2, 1) = Q3
**State: &(Q3, 0) = Q1
**State: &(Q1, 0) = Q3
**State: &(Q3, 0) = Q1
**State: &(Q1, 1) = Q0
**State: &(Q0, 1) = Q1
String not accepted.
```

Figura 17: Salida a archivo de texto de la opción 2.

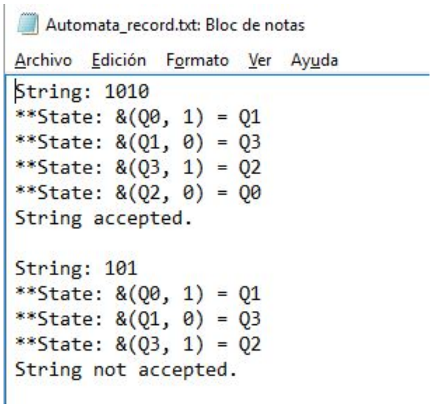
Si seleccionamos la opción 3, nos pedirá el nombre del archivo de texto desde el cual va a leer, y después desplegará el historial a consola y archivo:



```
Enter name of the file: read.txt
String: 1010
**State: &(Q0, 1) = Q1
**State: &(Q1, 0) = Q3
**State: &(Q3, 1) = Q2
**State: &(Q2, 0) = Q0
String accepted.

String: 101
**State: &(Q0, 1) = Q1
**State: &(Q1, 0) = Q3
**State: &(Q3, 1) = Q2
String not accepted.
```

Figura 18: Salida a consola de la opción 3.

Salida a archivo de texto:


Automata_record.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
String: 1010
**State: &(Q0, 1) = Q1
**State: &(Q1, 0) = Q3
**State: &(Q3, 1) = Q2
**State: &(Q2, 0) = Q0
String accepted.

String: 101
**State: &(Q0, 1) = Q1
**State: &(Q1, 0) = Q3
**State: &(Q3, 1) = Q2
String not accepted.
```

Figura 19: Salida a archivo de texto de la opción 3.

3.3. Código

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

#define STR_LENGTH 10000
#define Q0 0
#define Q1 1
#define Q2 2
#define Q3 3
#define TOP 100

int getString(int string_length, char string[string_length], int
    first_ch);
int getStringFile(int string_length, char string[string_length],
    int first_ch, FILE *read_fp);
bool automata(char *p, int mode);

int main (void)
{
        int option = 0, first_ch, last_ch = '␣', ch, top, i = 0,
            random, strings = 0, mode;
        char string[STR_LENGTH + 1], read_file[200];
        bool accepted;
        FILE *read_fp, *write_fp;

        srand((unsigned) time(NULL));

        printf("DFA␣recognizing␣the␣language:␣L␣=␣{w␣|␣w␣has␣both␣
            an␣even␣number␣of␣0's␣and␣an␣even␣number␣of␣1's}.\n");
        for(;;)
        {
                fflush(stdin);
                printf("**************MENU****************\n"
                "1.-Read␣from␣the␣command-line.\n"
                "2.-Generate␣a␣random␣string.\n"
                "3.-Read␣from␣a␣text␣file.\n"
                "4.-Exit.\n");
                printf("Option:␣");
                scanf("␣%d", &option);
                system("cls");
                switch(option)
                {
                case 1:
                fflush(stdin);
                last_ch = '␣';
                printf("Enter␣a␣string:␣");
                while((first_ch = getchar()) != '\n')
                {
                if (first_ch == '1' || first_ch == '0')
                        last_ch = getString(STR_LENGTH, string,
                            first_ch);
                if(last_ch == '\n')
                        break;
                }
```

```
automata(string, 'w');
break;

case 2:
top = rand() % TOP + 1;
printf("Random top: %d\n", top);
for(i = 0; i < top; i++)
{
if (rand() % 2)
        ch = '1';
else
        ch = '0';
string[i] = ch;
}
string[i] = '\0';
automata(string, 'w');
break;

case 3:
strings = 0;
for(i = 0; i < 200; i++)
        read_file[i] = ' ';
fflush(stdin);
                                        //SE ENCUENTRA
    AQU  DADO QUE EL BUFFER SE QUEDA CON EL '\n'
    QUE INGRESAMOS AL ELEGIR OPCI N
printf("Enter name of the file with the route i.e. 
    \"C:/Users/Jaime/Documents/read.txt\": ");
while((ch = getchar()) != '\n')
{
read_file[i++] = ch;
}
i = 0;
if ((read_fp = fopen(read_file, "rb")) == NULL)
{
fprintf(stderr, "Can't open: %s\n", read_file);
break;
}

while ((first_ch = getc(read_fp)) != EOF)
{
if(first_ch == '1' || first_ch == '0')
{
getStringFile(STR_LENGTH, string, first_ch, read_fp
    );
strings += 1;

if (strings == 1)
        mode = 'w';
else
        mode = 'a';

automata(string, mode);
}
}
break;
```

```
                case 4:
                        exit(EXIT_SUCCESS);
                break;
                }
        }
        return 0;
}

int getString(int string_length, char string[string_length], int
    first_ch)
{
        int i = 0; int ch;

        string[i++] = first_ch;

        while((ch = getchar()) == '1' || ch == '0' && i <
            string_length)
                string[i++] = ch;
        string[i] = '\0';

        return ch;
}

int getStringFile(int string_length, char string[string_length],
    int first_ch, FILE *read_fp)
{
        int i = 0; int ch;

        string[i++] = first_ch;

        while((ch = getc(read_fp)) == '1' || ch == '0' && i <
            string_length && ch != EOF)
                string[i++] = ch;
        string[i] = '\0';

        return ch;
}

bool automata(char *p, int mode)
{
        int state = Q0;
        bool move = false;
        FILE *fp;

        if (mode == 'w')
                fp = fopen("Automata_record.txt", "w");
        else
                fp = fopen("Automata_record.txt", "a");

        printf("String:␣%s\n", p);
        fprintf(fp, "String:␣%s\n", p);

        for(;;)
        {
                if (state == Q0 && *p == '1' && move == false)
                {
                state = Q1;
```

```
move = true;
printf(”**State:␣&(Q0,␣1)␣=␣Q1\n”);
fprintf(fp, ”**State:␣&(Q0,␣1)␣=␣Q1\n”);
}
else if (state == Q0 && *p == ’0’ && move == false)
{
state = Q2;
move = true;
printf(”**State:␣&(Q0,␣0)␣=␣Q2\n”);
fprintf(fp, ”**State:␣&(Q0,␣0)␣=␣Q2\n”);
}

if (state == Q1 && *p == ’1’ && move == false)
{
state = Q0;
move = true;
printf(”**State:␣&(Q1,␣1)␣=␣Q0\n”);
fprintf(fp, ”**State:␣&(Q1,␣1)␣=␣Q0\n”);
}
else if (state == Q1 && *p == ’0’ && move == false)
{
state = Q3;
move = true;
printf(”**State:␣&(Q1,␣0)␣=␣Q3\n”);
fprintf(fp, ”**State:␣&(Q1,␣0)␣=␣Q3\n”);
}

if (state == Q2 && *p == ’0’ && move == false)
{
state = Q0;
move = true;
printf(”**State:␣&(Q2,␣0)␣=␣Q0\n”);
fprintf(fp, ”**State:␣&(Q2,␣0)␣=␣Q0\n”);
}
else if (state == Q2 && *p == ’1’ && move == false)
{
state = Q3;
move = true;
printf(”**State:␣&(Q2,␣1)␣=␣Q3\n”);
fprintf(fp, ”**State:␣&(Q2,␣1)␣=␣Q3\n”);
}

if (state == Q3 && *p == ’0’ && move == false)
{
state = Q1;
move = true;
printf(”**State:␣&(Q3,␣0)␣=␣Q1\n”);
fprintf(fp, ”**State:␣&(Q3,␣0)␣=␣Q1\n”);
}
else if (state == Q3 && *p == ’1’ && move == false)
{
state = Q2;
move = true;
printf(”**State:␣&(Q3,␣1)␣=␣Q2\n”);
fprintf(fp, ”**State:␣&(Q3,␣1)␣=␣Q2\n”);
}
```

```
                if(state == Q0 && *p == '\0')
                {
                printf("String␣accepted.\n\n");
                fprintf(fp, "String␣accepted.\n\n");
                return true;
                }
                else if (state != Q0 && *p == '\0')
                {
                printf("String␣not␣accepted.\n\n");
                fprintf(fp, "String␣not␣accepted.\n\n");
                return false;
                }

                p++;
                move = false;
        }
        fclose(fp);
}
```

4. Práctica 5: Protocolo

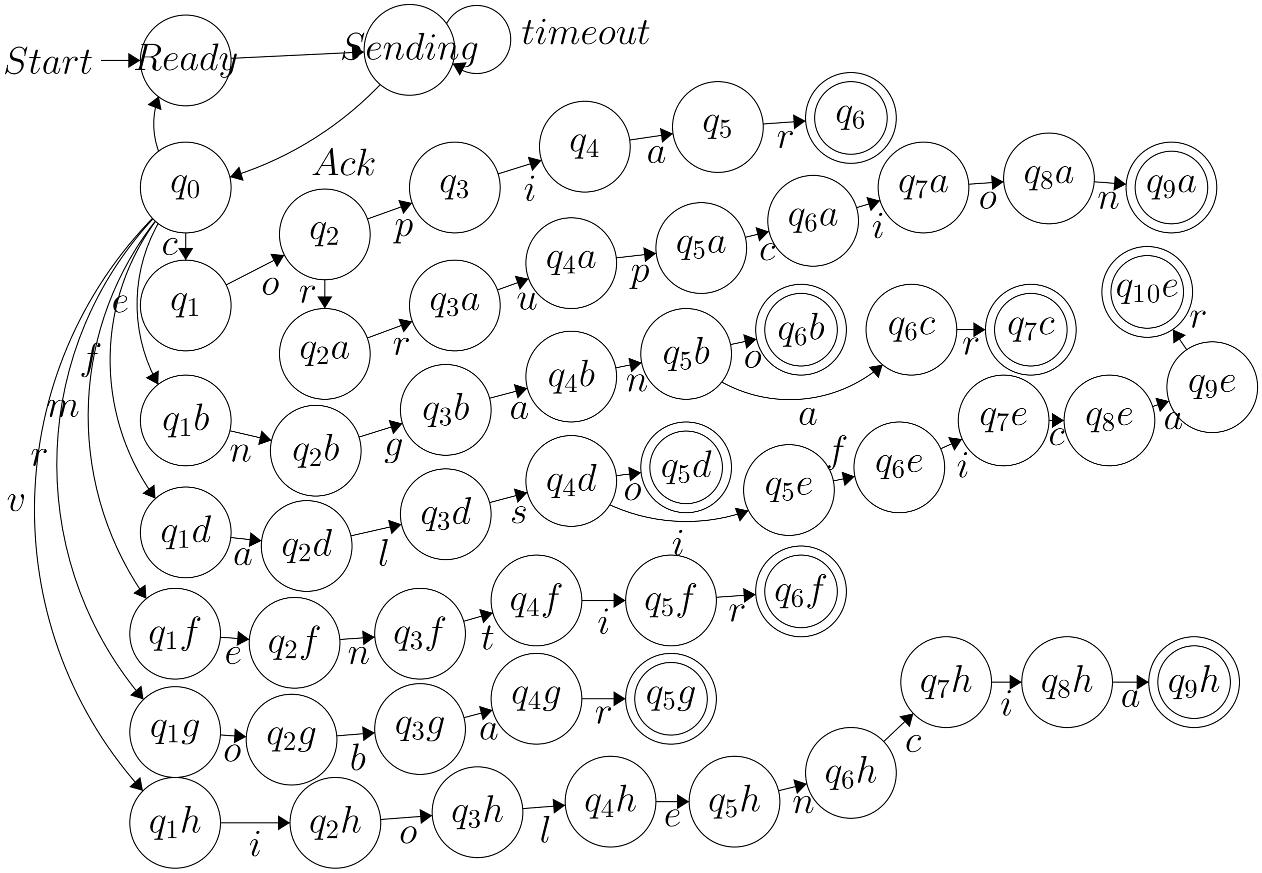
4.1. Descripción

Este programa simula un protocolo de envío de datos, recibe bloques de 100 carácteres que lee desde un archivo de texto los cuales retiene por 3 segundos y luego manda bloque por bloque hacia el .^{a}cknowledge"quien se encarga de reconocer las siguientes palabras: copiar, corrupción, engaño, engañar, falso, falsificar, mentir, robar, violencia, traficar.

Después de analizar cada bloque regresa el inicio y en el caso de la simulación existe la posibilidad de que el protocolo se haya apagado causando que el programa deje de leer más bloques y se detenga, por otro lado en la opción .^{A}lways On", el programa no acabará hasta que haya leído todo el archivo de texto.

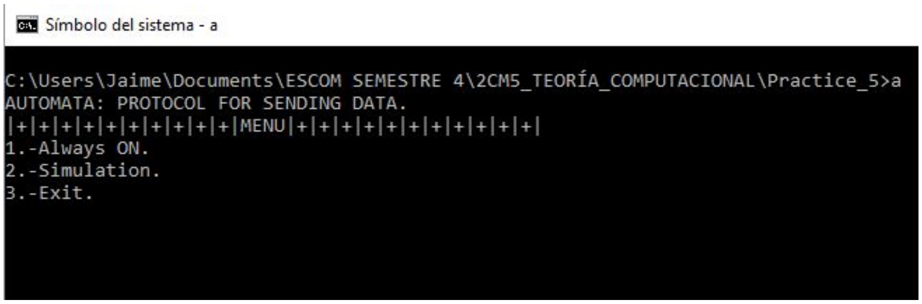
Al terminar el programa desplegará el historial a consola y a un archivo de texto de como trabajó cada autómata indicando las palabras sospechosas y el número de veces que vió cada una.

Grafo del Protocolo:



4.2. Ejecución

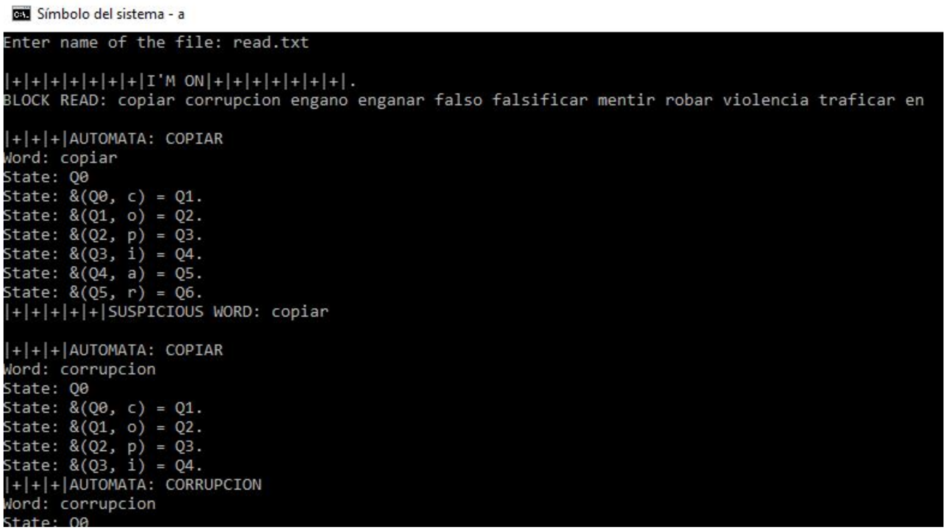
Al ejecutar el programa se nos presenta el siguiente menú:



```
C:\Users\Jaime\Documents\ESCOM SEMESTRE 4\2CM5_TEORÍA_COMPUTACIONAL\Practice_5>a
AUTOMATA: PROTOCOL FOR SENDING DATA.
|+|+|+|+|+|+|+|+|+|+|MENU|+|+|+|+|+|+|+|+|+|+|
1.-Always ON.
2.-Simulation.
3.-Exit.
```

Figura 20: Menú Principal.

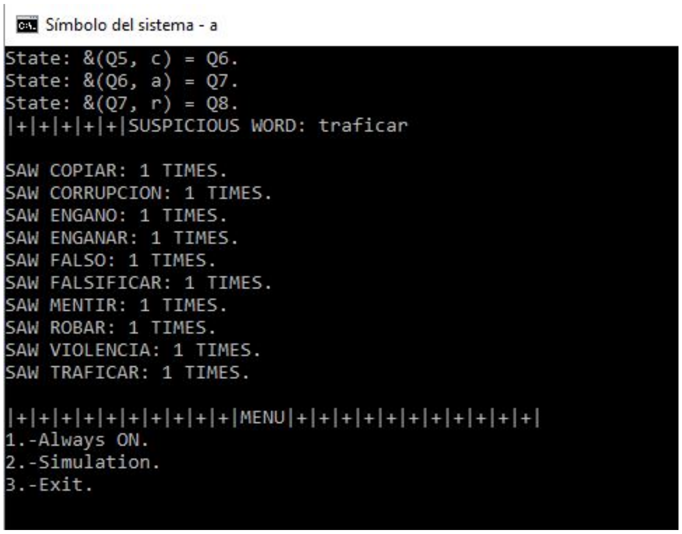
Si seleccionamos la opción 1 el programa nos pedirá el nombre del archivo a leer, y posteriormente nos mostrará el bloque de 100 carácteres que leyó, esperará 3 segundos:



```
Enter name of the file: read.txt

|+|+|+|+|+|+|+|I'M ON|+|+|+|+|+|+|+|.
BLOCK READ: copiar corrupcion engano enganar falso falsificar mentir robar violencia traficar en

|+|+|+|AUTOMATA: COPIAR
Word: copiar
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
State: &(Q2, p) = Q3.
State: &(Q3, i) = Q4.
State: &(Q4, a) = Q5.
State: &(Q5, r) = Q6.
|+|+|+|+|+|SUSPICIOUS WORD: copiar

|+|+|+|AUTOMATA: COPIAR
Word: corrupcion
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
State: &(Q2, p) = Q3.
State: &(Q3, i) = Q4.
|+|+|+|AUTOMATA: CORRUPCION
Word: corrupcion
State: Q0
```

Figura 21: Salida a consola opción 1, bloque e historial.

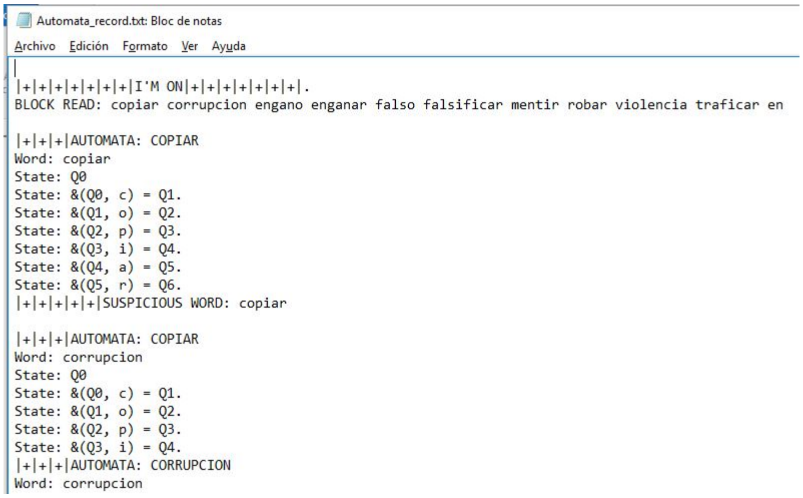
A continuación desplegará el historial de como trabajó el acknowledge, al final nos mostrará el número de veces que vió cada palabra:



```
State: &(Q5, c) = Q6.
State: &(Q6, a) = Q7.
State: &(Q7, r) = Q8.
|+|+|+|+|+|SUSPICIOUS WORD: traficar

SAW COPIAR: 1 TIMES.
SAW CORRUPCION: 1 TIMES.
SAW ENGANO: 1 TIMES.
SAW ENGANAR: 1 TIMES.
SAW FALSO: 1 TIMES.
SAW FALSIFICAR: 1 TIMES.
SAW MENTIR: 1 TIMES.
SAW ROBAR: 1 TIMES.
SAW VIOLENCIA: 1 TIMES.
SAW TRAFICAR: 1 TIMES.

|+|+|+|+|+|+|+|+|+|+|MENU|+|+|+|+|+|+|+|+|+|+|
1.-Always ON.
2.-Simulation.
3.-Exit.
```

Figura 22: Salida a consola opción 1, bloque e historial.

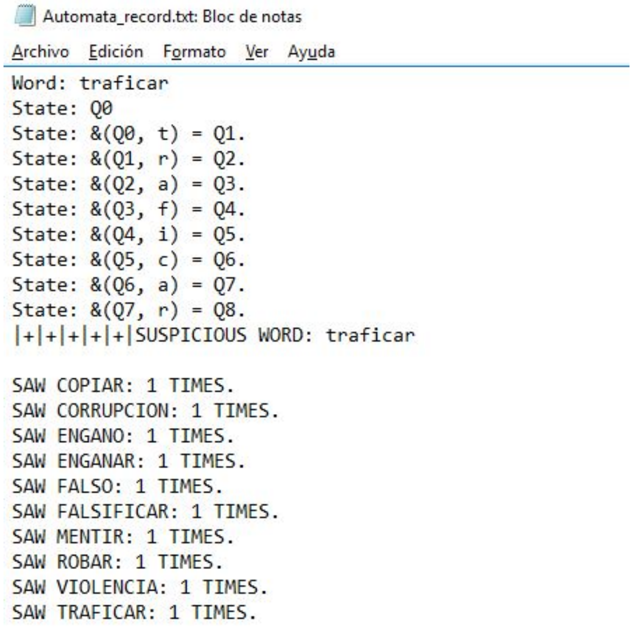
Salida a archivo de texto:



```
|+|+|+|+|+|+|+|I'M ON|+|+|+|+|+|+|+|.
BLOCK READ: copiar corrupcion engano enganar falso falsificar mentir robar violencia traficar en

|+|+|+|AUTOMATA: COPIAR
Word: copiar
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
State: &(Q2, p) = Q3.
State: &(Q3, i) = Q4.
State: &(Q4, a) = Q5.
State: &(Q5, r) = Q6.
|+|+|+|+|+|SUSPICIOUS WORD: copiar

|+|+|+|AUTOMATA: COPIAR
Word: corrupcion
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
State: &(Q2, p) = Q3.
State: &(Q3, i) = Q4.
|+|+|+|AUTOMATA: CORRUPCION
Word: corrupcion
```

Figura 23: Salida archivo de texto opción 1, bloque e historial.

Salida a archivo de texto:

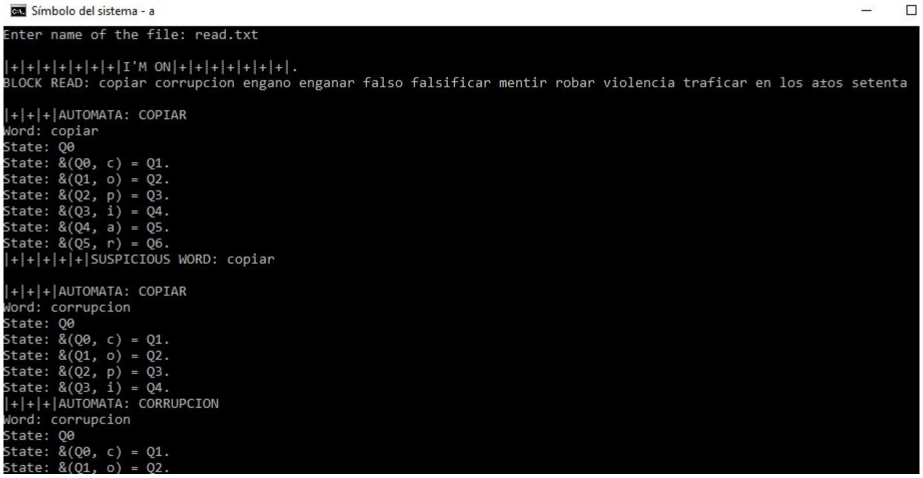


Automata_record.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
Word: traficar
State: Q0
State: &(Q0, t) = Q1.
State: &(Q1, r) = Q2.
State: &(Q2, a) = Q3.
State: &(Q3, f) = Q4.
State: &(Q4, i) = Q5.
State: &(Q5, c) = Q6.
State: &(Q6, a) = Q7.
State: &(Q7, r) = Q8.
|+|+|+|+|+|SUSPICIOUS WORD: traficar

SAW COPIAR: 1 TIMES.
SAW CORRUPCION: 1 TIMES.
SAW ENGANO: 1 TIMES.
SAW ENGANAR: 1 TIMES.
SAW FALSO: 1 TIMES.
SAW FALSIFICAR: 1 TIMES.
SAW MENTIR: 1 TIMES.
SAW ROBAR: 1 TIMES.
SAW VIOLENCIA: 1 TIMES.
SAW TRAFICAR: 1 TIMES.
```

Figura 24: Salida a archivo de texto opción 1, bloque e historial.

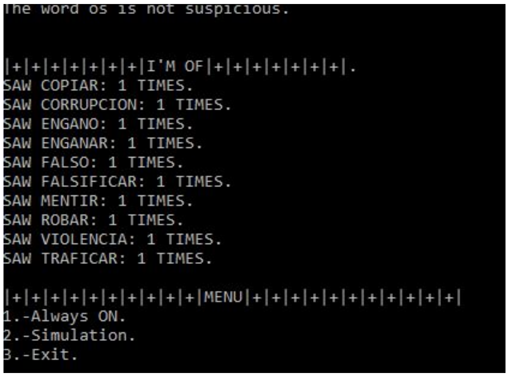
Si seleccionamos la opción 2 ”Simulation.$^{\text{el}}$ programa nos pedirá el nombre del archivo de texto a leer y siempre leerá el primer bloque, después será aleatorio si el protocolo seguirá encendido o no, indicándonos con un Ï’M ONçuando esté encendido y un Ï’M OFF.$^{\text{en}}$ caso contrario:



```
Enter name of the file: read.txt

|+|+|+|+|+|+|+|I'M ON|+|+|+|+|+|+|+|.
BLOCK READ: copiar corrupcion engano enganar falso falsificar mentir robar violencia traficar en los a±os setenta

|+|+|+|AUTOMATA: COPIAR
Word: copiar
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
State: &(Q2, p) = Q3.
State: &(Q3, i) = Q4.
State: &(Q4, a) = Q5.
State: &(Q5, r) = Q6.
|+|+|+|+|+|SUSPICIOUS WORD: copiar

|+|+|+|AUTOMATA: COPIAR
Word: corrupcion
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
State: &(Q2, p) = Q3.
State: &(Q3, i) = Q4.
|+|+|+|AUTOMATA: CORRUPCION
Word: corrupcion
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
```

Figura 25: Salida a consola opción 2, bloque e historial.

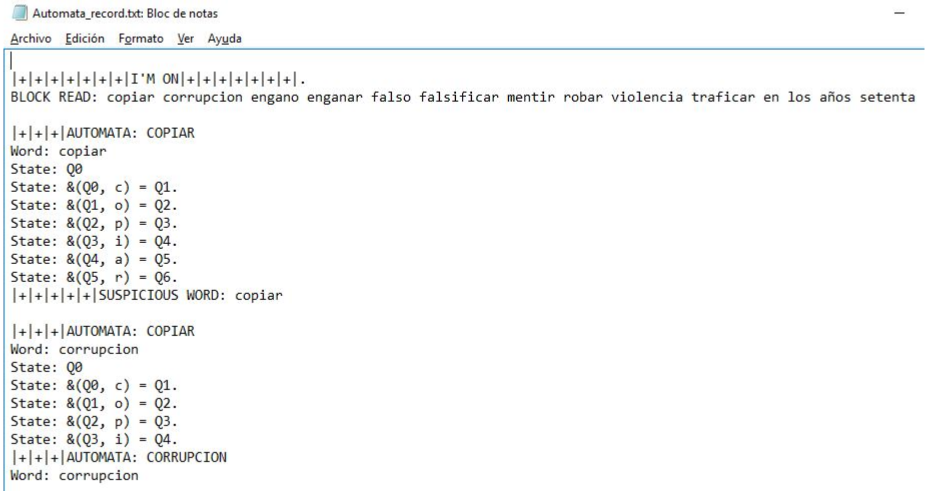
En este caso, por ejemplo, podemos observar que el protocolo se apagó después de leer el primer bloque:



```
The word os is not suspicious.


|+|+|+|+|+|+|+|I'M OF|+|+|+|+|+|+|+|.
SAW COPIAR: 1 TIMES.
SAW CORRUPCION: 1 TIMES.
SAW ENGANO: 1 TIMES.
SAW ENGANAR: 1 TIMES.
SAW FALSO: 1 TIMES.
SAW FALSIFICAR: 1 TIMES.
SAW MENTIR: 1 TIMES.
SAW ROBAR: 1 TIMES.
SAW VIOLENCIA: 1 TIMES.
SAW TRAFICAR: 1 TIMES.

|+|+|+|+|+|+|+|+|+|+|MENU|+|+|+|+|+|+|+|+|+|+|
1.-Always ON.
2.-Simulation.
3.-Exit.
```

Figura 26: Salida a consola opción 2, bloque e historial.

Salida a archivo de texto:

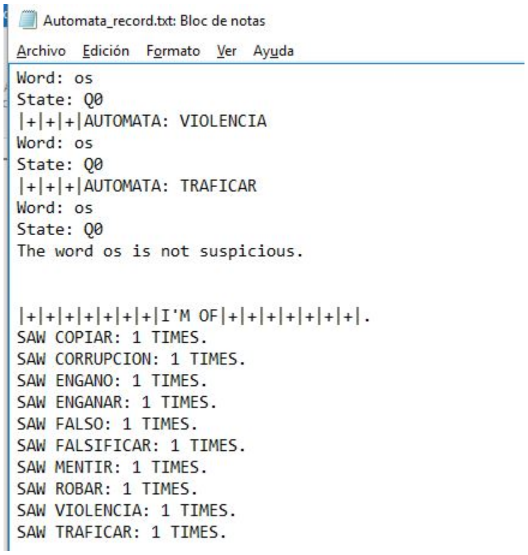


Automata_record.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
|+|+|+|+|+|+|+|I'M ON|+|+|+|+|+|+|+|.
BLOCK READ: copiar corrupcion engano enganar falso falsificar mentir robar violencia traficar en los años setenta

|+|+|+|AUTOMATA: COPIAR
Word: copiar
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
State: &(Q2, p) = Q3.
State: &(Q3, i) = Q4.
State: &(Q4, a) = Q5.
State: &(Q5, r) = Q6.
|+|+|+|+|+|SUSPICIOUS WORD: copiar

|+|+|+|AUTOMATA: COPIAR
Word: corrupcion
State: Q0
State: &(Q0, c) = Q1.
State: &(Q1, o) = Q2.
State: &(Q2, p) = Q3.
State: &(Q3, i) = Q4.
|+|+|+|AUTOMATA: CORRUPCION
Word: corrupcion
```

Figura 27: Salida a archivo de texto opción 2, bloque e historial.

Salida a archivo de texto:



Automata_record.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda

```
Word: os
State: Q0
|+|+|+|AUTOMATA: VIOLENCIA
Word: os
State: Q0
|+|+|+|AUTOMATA: TRAFICAR
Word: os
State: Q0
The word os is not suspicious.


|+|+|+|+|+|+|+|I'M OF|+|+|+|+|+|+|+|.
SAW COPIAR: 1 TIMES.
SAW CORRUPCION: 1 TIMES.
SAW ENGANO: 1 TIMES.
SAW ENGANAR: 1 TIMES.
SAW FALSO: 1 TIMES.
SAW FALSIFICAR: 1 TIMES.
SAW MENTIR: 1 TIMES.
SAW ROBAR: 1 TIMES.
SAW VIOLENCIA: 1 TIMES.
SAW TRAFICAR: 1 TIMES.
```

Figura 28: Salida a archivo de texto opción 2, bloque e historial.

4.3. Código

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

#define BLOCK_LENGTH 100
#define STR_LENGTH 100
#define Q0 0
#define Q1 1
#define Q2 2
#define Q3 3
#define Q4 4
#define Q5 5
#define Q6 6
#define Q7 7
#define Q8 8
#define Q9 9
#define Q10 10

bool getBlock(char block[], int n, FILE *read_fp);
int getString(char block[], char string[]);
int acknowledge(char *p, FILE *write_fp);
bool automCopiar(char *p, FILE *write_fp);
bool automCorrupcion(char *p, FILE *write_fp);
bool automEngano(char *p, FILE *write_fp);
bool automEnganar(char *p, FILE *write_fp);
bool automFalso(char *p, FILE *write_fp);
bool automFalsificar(char *p, FILE *write_fp);
bool automMentir(char *p, FILE *write_fp);
bool automRobar(char *p, FILE *write_fp);
bool automViolencia(char *p, FILE *write_fp);
bool automTraficar(char *p, FILE *write_fp);

int block_ch = 0;

int main (void)
{
        int option = 0, i = 0, ch, characters_read = 0, last_ch = '
            ␣', on = 1;
        char block[BLOCK_LENGTH + 1], read_file[200] = {'␣'},
            string[STR_LENGTH + 1];
        bool keep_reading = true;
        FILE *read_fp, *write_fp;
        int copiar = 0, corrupcion = 0, engano = 0, enganar = 0,
            falso = 0, falsificar = 0, mentir = 0, robar = 0,
            violencia = 0, traficar = 0;

        srand((unsigned) time (NULL));

        printf(”AUTOMATA:␣PROTOCOL␣FOR␣SENDING␣DATA.\n”);
        for(;;)
        {
```

```
printf("|+|+|+|+|+|+|+|+|+|+|MENU
     |+|+|+|+|+|+|+|+|+|+|+|\n1.-Always␣ON.\n2.-
     Simulation.\n3.-Exit.\n");
scanf("␣%d", &option);
system("cls");
switch(option)
{
case 1:
for(i = 0; i < 200; i++)
        read_file[i] = '␣';
i = 0;
fflush(stdin);
printf("Enter␣name␣of␣the␣file␣with␣the␣route␣i.e.␣
     \"C:/Users/Jaime/Documents/read.txt\":␣");
while((ch = getchar()) != '\n')
        read_file[i++] = ch;
if ((read_fp = fopen(read_file, "rb")) == NULL)
{
fprintf(stderr, "Can't␣open:␣%s\n", read_file);
break;
}
if ((write_fp = fopen("Automata_record.txt", "w"))
     == NULL)
        fprintf(stderr, "Can't␣write␣to␣file.\n");
while(keep_reading)
{
printf("\n|+|+|+|+|+|+|+|I'M␣ON|+|+|+|+|+|+|+|.\n")
     ;
fprintf(write_fp, "\n|+|+|+|+|+|+|+|I'M␣ON
     |+|+|+|+|+|+|+|.\n");
keep_reading = getBlock(block, BLOCK_LENGTH,
     read_fp);
printf("BLOCK_READ:␣%s\n\n", block);
fprintf(write_fp, "BLOCK_READ:␣%s\n\n", block);
Sleep(3000);
while(getString(block, string) != '\0')
{
switch(acknowledge(string, write_fp))
{
case 1:
        copiar += 1;
break;
case 2:
        corrupcion += 1;
break;
case 3:
        engano += 1;
break;
case 4:
        enganar += 1;
break;
case 5:
        falso += 1;
break;
case 6:
        falsificar += 1;
break;
case 7:
        mentir += 1;
break;
case 8:
        robar += 1;
break;
case 9:
        violencia += 1;
break;
case 10:
        traficar += 1;
break;
case 11:
        printf("The␣word␣%s␣is␣not␣suspicious.\n\n"
            , string);
        fprintf(write_fp, "The␣word␣%s␣is␣not␣
            suspicious.\n\n", string);
break;
default:
        printf("There␣was␣an␣error.\n");
        fprintf(write_fp, "There␣was␣an␣error.\n");
break;
}
}
block_ch = 0;
}
keep_reading = true;
fclose(read_fp);
printf("SAW␣COPIAR:␣%d␣TIMES.\nSAW␣CORRUPCION:␣%d␣
    TIMES.\nSAW␣ENGANO:␣%d␣TIMES.\nSAW␣ENGANAR:␣%d␣
    TIMES.\nSAW␣FALSO:␣%d␣TIMES.\nSAW␣FALSIFICAR:␣%
    d␣TIMES.\nSAW␣MENTIR:␣%d␣TIMES.\nSAW␣ROBAR:␣%d␣
    TIMES.\nSAW␣VIOLENCIA:␣%d␣TIMES.\nSAW␣TRAFICAR:
    ␣%d␣TIMES.\n\n", copiar, corrupcion, engano,
    enganar, falso, falsificar, mentir, robar,
    violencia, traficar);
fprintf(write_fp, "SAW␣COPIAR:␣%d␣TIMES.\nSAW␣
    CORRUPCION:␣%d␣TIMES.\nSAW␣ENGANO:␣%d␣TIMES.\
    nSAW␣ENGANAR:␣%d␣TIMES.\nSAW␣FALSO:␣%d␣TIMES.\
    nSAW␣FALSIFICAR:␣%d␣TIMES.\nSAW␣MENTIR:␣%d␣
    TIMES.\nSAW␣ROBAR:␣%d␣TIMES.\nSAW␣VIOLENCIA:␣%d
    ␣TIMES.\nSAW␣TRAFICAR:␣%d␣TIMES.\n\n", copiar,
    corrupcion, engano, enganar, falso, falsificar,
     mentir, robar, violencia, traficar);
fclose(write_fp);
copiar = 0; corrupcion = 0; engano = 0; enganar =
    0; falso = 0; falsificar = 0; mentir = 0; robar
     = 0; violencia = 0; traficar = 0;
break;
case 2:
for(i = 0; i < 200; i++)
        read_file[i] = '␣';
i = 0;
fflush(stdin);
printf("Enter␣name␣of␣the␣file␣with␣the␣route␣i.e.␣
    \"C:/Users/Jaime/Documents/read.txt\":␣");
while((ch = getchar()) != '\n')
```

```
        read_file[i++] = ch;
if ((read_fp = fopen(read_file, "rb")) == NULL)
{
        fprintf(stderr, "Can't␣open:␣%s\n",
            read_file);
        break;
}
if ((write_fp = fopen("Automata_record.txt", "w"))
    == NULL)
        fprintf(stderr, "Can't␣write␣to␣file.\n");
while(keep_reading && on == 1)
{
on = rand() % 2;
printf("\n|+|+|+|+|+|+|+|I'M␣ON|+|+|+|+|+|+|+|.\n")
    ;
fprintf(write_fp, "\n|+|+|+|+|+|+|+|I'M␣ON
    |+|+|+|+|+|+|+|.\n");
keep_reading = getBlock(block, BLOCK_LENGTH,
    read_fp);
printf("BLOCK_READ:␣%s\n\n", block);
fprintf(write_fp, "BLOCK_READ:␣%s\n\n", block);
Sleep(3000);
while((ch = getString(block, string)) != '\0')
{
switch(acknowledge(string, write_fp))
{
case 1:
        copiar += 1;
break;
case 2:
        corrupcion += 1;
break;
case 3:
        engano += 1;
break;
case 4:
        enganar += 1;
break;
case 5:
        falso += 1;
break;
case 6:
        falsificar += 1;
break;
case 7:
        mentir += 1;
break;
case 8:
        robar += 1;
break;
case 9:
        violencia += 1;
break;
case 10:
        traficar += 1;
break;
case 11:
```

```
                        printf("The␣word␣%s␣is␣not␣suspicious.\n\n"
                            , string);
                        fprintf(write_fp, "The␣word␣%s␣is␣not␣
                            suspicious.\n\n", string);
                break;
                default:
                        printf("There␣was␣an␣error.\n");
                        fprintf(write_fp, "There␣was␣an␣error.\n");
                break;
                }
                }
                block_ch = 0;
                }
                on = 1;
                keep_reading = true;
                fclose(read_fp);
                printf("\n|+|+|+|+|+|+|+|I'M␣OF|+|+|+|+|+|+|+|.\n")
                    ;
                fprintf(write_fp, "\n|+|+|+|+|+|+|+|I'M␣OF
                    |+|+|+|+|+|+|+|.\n");
                printf("SAW␣COPIAR:␣%d␣TIMES.\nSAW␣CORRUPCION:␣%d␣
                    TIMES.\nSAW␣ENGANO:␣%d␣TIMES.\nSAW␣ENGANAR:␣%d␣
                    TIMES.\nSAW␣FALSO:␣%d␣TIMES.\nSAW␣FALSIFICAR:␣%
                    d␣TIMES.\nSAW␣MENTIR:␣%d␣TIMES.\nSAW␣ROBAR:␣%d␣
                    TIMES.\nSAW␣VIOLENCIA:␣%d␣TIMES.\nSAW␣TRAFICAR:
                    ␣%d␣TIMES.\n\n", copiar, corrupcion, engano,
                    enganar, falso, falsificar, mentir, robar,
                    violencia, traficar);
                fprintf(write_fp, "SAW␣COPIAR:␣%d␣TIMES.\nSAW␣
                    CORRUPCION:␣%d␣TIMES.\nSAW␣ENGANO:␣%d␣TIMES.\
                    nSAW␣ENGANAR:␣%d␣TIMES.\nSAW␣FALSO:␣%d␣TIMES.\
                    nSAW␣FALSIFICAR:␣%d␣TIMES.\nSAW␣MENTIR:␣%d␣
                    TIMES.\nSAW␣ROBAR:␣%d␣TIMES.\nSAW␣VIOLENCIA:␣%d
                    ␣TIMES.\nSAW␣TRAFICAR:␣%d␣TIMES.\n\n", copiar,
                    corrupcion, engano, enganar, falso, falsificar,
                     mentir, robar, violencia, traficar);
                fclose(write_fp);
                copiar = 0; corrupcion = 0; engano = 0; enganar =
                    0; falso = 0; falsificar = 0; mentir = 0; robar
                     = 0; violencia = 0; traficar = 0;
                break;
                case 3:
                        exit(EXIT_SUCCESS);
                break;
                default:
                        printf("Choose␣a␣correct␣option.\n");
                break;
                }
        }
        return 0;
}

bool getBlock(char block[], int n, FILE *read_fp)
{
        int ch, i = 0;
        long file_pos;
        while(i <= n)
        {
                ch = getc(read_fp);
                if(ch == EOF)
                {
                        block[i] = '\0';
                        return false;
                }
                else if (ch == '␣')
                        file_pos = ftell(read_fp);
                block[i++] = ch;
        }
        if ((ch = getc(read_fp)) != '␣')
                fseek(read_fp, file_pos, SEEK_SET);
        block[i] = '\0';

        return true;
}

int getString(char block[], char string[])
{
        int i = 0;

        while(block[block_ch] != '␣' && ((block[block_ch] >= 65 &&
            block[block_ch] <= 90) || (block[block_ch] >= 97 &&
            block[block_ch] <= 122)) && block[block_ch] != '\0')
                string[i++] = block[block_ch++];

        string[i] = '\0';
        return block[block_ch++];
}

int acknowledge(char *p, FILE *write_fp)
{
        if (automCopiar(p, write_fp))
                return 1;
        else if (automCorrupcion(p, write_fp))
                return 2;
        else if (automEngano(p, write_fp))
                return 3;
        else if (automEnganar(p, write_fp))
                return 4;
        else if (automFalso(p, write_fp))
                return 5;
        else if (automFalsificar(p, write_fp))
                return 6;
        else if (automMentir(p, write_fp))
                return 7;
        else if (automRobar(p, write_fp))
                return 8;
        else if (automViolencia(p, write_fp))
                return 9;
        else if (automTraficar(p, write_fp))
                return 10;
        else
                return 11;
}
```

```
bool automCopiar(char *p, FILE *write_fp)
{
        int state = Q0;
        char *word = p;

        printf("|+|+|+|AUTOMATA:␣COPIAR\nWord:␣%s\n", p);
        printf("State:␣Q0\n");
        fprintf(write_fp, "|+|+|+|AUTOMATA:␣COPIAR\nWord:␣%s\n", p)
            ;
        fprintf(write_fp, "State:␣Q0\n");
        while(*p != '\0')
        {
                //AUTOMATA CORE
                if(toupper(*p) == 'C' && state == Q0)
                {
                        state = Q1;
                        printf("State:␣&(Q0,␣c)␣=␣Q1.\n");
                        fprintf(write_fp, "State:␣&(Q0,␣c)␣=␣Q1.\n"
                            );
                }
                else if(toupper(*p) == 'O' && state == Q1)
                {
                        state = Q2;
                        printf("State:␣&(Q1,␣o)␣=␣Q2.\n");
                        fprintf(write_fp, "State:␣&(Q1,␣o)␣=␣Q2.\n"
                            );
                }
                else if(toupper(*p) == 'P' && state == Q2)
                {
                        state = Q3;
                        printf("State:␣&(Q2,␣p)␣=␣Q3.\n");
                        fprintf(write_fp, "State:␣&(Q2,␣p)␣=␣Q3.\n"
                            );
                }
                else if(toupper(*p) == 'I' && state == Q3)
                {
                        state = Q4;
                        printf("State:␣&(Q3,␣i)␣=␣Q4.\n");
                        fprintf(write_fp, "State:␣&(Q3,␣i)␣=␣Q4.\n"
                            );
                }
                else if(toupper(*p) == 'A' && state == Q4)
                {
                        state = Q5;
                        printf("State:␣&(Q4,␣a)␣=␣Q5.\n");
                        fprintf(write_fp, "State:␣&(Q4,␣a)␣=␣Q5.\n"
                            );
                }
                else if(toupper(*p) == 'R' && state == Q5)
                {
                        state = Q6;
                        printf("State:␣&(Q5,␣r)␣=␣Q6.\n");
                        fprintf(write_fp, "State:␣&(Q5,␣r)␣=␣Q6.\n"
                            );
                }
                //FINAL STATE
                if(state == Q6)
                {
                printf("|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n\n", word);
                fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n
                    \n", word);
                return true;
                }

                p++;
        }
        return false;
}

bool automCorrupcion(char *p, FILE *write_fp)
{
        int state = Q0;
        char *word = p;

        printf("|+|+|+|AUTOMATA:␣CORRUPCION\nWord:␣%s\n", p);
        printf("State:␣Q0\n");
        fprintf(write_fp, "|+|+|+|AUTOMATA:␣CORRUPCION\nWord:␣%s\n"
            , p);
        fprintf(write_fp, "State:␣Q0\n");
        while(*p != '\0')
        {
                //AUTOMATA CORE
                if(toupper(*p) == 'C' && state == Q0)
                {
                        state = Q1;
                        printf("State:␣&(Q0,␣c)␣=␣Q1.\n");
                        fprintf(write_fp, "State:␣&(Q0,␣c)␣=␣Q1.\n"
                            );
                }
                else if(toupper(*p) == 'O' && state == Q1)
                {
                        state = Q2;
                        printf("State:␣&(Q1,␣o)␣=␣Q2.\n");
                        fprintf(write_fp, "State:␣&(Q1,␣o)␣=␣Q2.\n"
                            );
                }
                else if(toupper(*p) == 'R' && state == Q2)
                {
                        state = Q3;
                        printf("State:␣&(Q2,␣r)␣=␣Q3.\n");
                        fprintf(write_fp, "State:␣&(Q2,␣r)␣=␣Q3.\n"
                            );
                }
                else if(toupper(*p) == 'R' && state == Q3)
                {
                        state = Q4;
                        printf("State:␣&(Q3,␣r)␣=␣Q4.\n");
                        fprintf(write_fp, "State:␣&(Q3,␣r)␣=␣Q4.\n"
                            );
                }
                else if(toupper(*p) == 'U' && state == Q4)
                {
                        state = Q5;
                        printf(”State:␣&(Q4,␣u)␣=␣Q5.\n”);
                        fprintf(write_fp, ”State:␣&(Q4,␣u)␣=␣Q5.\n”
                            );
                }
                else if(toupper(*p) == ’P’ && state == Q5)
                {
                        state = Q6;
                        printf(”State:␣&(Q5,␣p)␣=␣Q6.\n”);
                        fprintf(write_fp, ”State:␣&(Q5,␣p)␣=␣Q6.\n”
                            );
                }
                else if(toupper(*p) == ’C’ && state == Q6)
                {
                        state = Q7;
                        printf(”State:␣&(Q6,␣c)␣=␣Q7.\n”);
                        fprintf(write_fp, ”State:␣&(Q6,␣c)␣=␣Q7.\n”
                            );
                }
                else if(toupper(*p) == ’I’ && state == Q7)
                {
                        state = Q8;
                        printf(”State:␣&(Q7,␣i)␣=␣Q8.\n”);
                        fprintf(write_fp, ”State:␣&(Q7,␣i)␣=␣Q8.\n”
                            );
                }
                else if(toupper(*p) == ’O’ && state == Q8)
                {
                        state = Q9;
                        printf(”State:␣&(Q8,␣o)␣=␣Q9.\n”);
                        fprintf(write_fp, ”State:␣&(Q8,␣o)␣=␣Q9.\n”
                            );
                }
                else if(toupper(*p) == ’N’ && state == Q9)
                {
                        state = Q10;
                        printf(”State:␣&(Q9,␣n)␣=␣Q10.\n”);
                        fprintf(write_fp, ”State:␣&(Q9,␣n)␣=␣Q10.\n
                            ”);
                }
                //FINAL STATE
                if(state == Q10)
                {
                printf(”|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n\n”, word);
                fprintf(write_fp, ”|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n
                    \n”, word);
                return true;
                }

                p++;
        }
        return false;
}

bool automEngano(char *p, FILE *write_fp)
{
        int state = Q0;
```

```
char *word = p;

printf("|+|+|+|AUTOMATA:␣ENGANO\nWord:␣%s\n", p);
printf("State:␣Q0\n");
fprintf(write_fp, "|+|+|+|AUTOMATA:␣ENGANO\nWord:␣%s\n", p)
    ;
fprintf(write_fp, "State:␣Q0\n");
while(*p != '\0')
{
        //AUTOMATA CORE
        if(toupper(*p) == 'E' && state == Q0)
        {
                state = Q1;
                printf("State:␣&(Q0,␣e)␣=␣Q1.\n");
                fprintf(write_fp, "State:␣&(Q0,␣e)␣=␣Q1.\n"
                    );
        }
        else if(toupper(*p) == 'N' && state == Q1)
        {
                state = Q2;
                printf("State:␣&(Q1,␣n)␣=␣Q2.\n");
                fprintf(write_fp, "State:␣&(Q1,␣n)␣=␣Q2.\n"
                    );
        }
        else if(toupper(*p) == 'G' && state == Q2)
        {
                state = Q3;
                printf("State:␣&(Q2,␣g)␣=␣Q3.\n");
                fprintf(write_fp, "State:␣&(Q2,␣g)␣=␣Q3.\n"
                    );
        }
        else if(toupper(*p) == 'A' && state == Q3)
        {
                state = Q4;
                printf("State:␣&(Q3,␣a)␣=␣Q4.\n");
                fprintf(write_fp, "State:␣&(Q3,␣a)␣=␣Q4.\n"
                    );
        }
        else if(toupper(*p) == 'N' && state == Q4)
        {
                state = Q5;
                printf("State:␣&(Q4,␣n)␣=␣Q5.\n");
                fprintf(write_fp, "State:␣&(Q4,␣n)␣=␣Q5.\n"
                    );
        }
        else if(toupper(*p) == 'O' && state == Q5)
        {
                state = Q6;
                printf("State:␣&(Q5,␣o)␣=␣Q6.\n");
                fprintf(write_fp, "State:␣&(Q5,␣o)␣=␣Q6.\n"
                    );
        }
        //FINAL STATE
        if(state == Q6)
        {
        printf("|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n\n", word);
```

```
                        fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n
                            \n", word);
                        return true;
                        }

                        p++;
            }
            return false;
}

bool automEnganar(char *p, FILE *write_fp)
{
            int state = Q0;
            char *word = p;

            printf("|+|+|+|AUTOMATA:␣ENGANAR\nWord:␣%s\n", p);
            printf("State:␣Q0\n");
            fprintf(write_fp, "|+|+|+|AUTOMATA:␣ENGANAR\nWord:␣%s\n", p
                );
            fprintf(write_fp, "State:␣Q0\n");
            while(*p != '\0')
            {
                        //AUTOMATA CORE
                        if(toupper(*p) == 'E' && state == Q0)
                        {
                                    state = Q1;
                                    printf("State:␣&(Q0,␣e)␣=␣Q1.\n");
                                    fprintf(write_fp, "State:␣&(Q0,␣e)␣=␣Q1.\n"
                                        );
                        }
                        else if(toupper(*p) == 'N' && state == Q1)
                        {
                                    state = Q2;
                                    printf("State:␣&(Q1,␣n)␣=␣Q2.\n");
                                    fprintf(write_fp, "State:␣&(Q1,␣n)␣=␣Q2.\n"
                                        );
                        }
                        else if(toupper(*p) == 'G' && state == Q2)
                        {
                                    state = Q3;
                                    printf("State:␣&(Q2,␣g)␣=␣Q3.\n");
                                    fprintf(write_fp, "State:␣&(Q2,␣g)␣=␣Q3.\n"
                                        );
                        }
                        else if(toupper(*p) == 'A' && state == Q3)
                        {
                                    state = Q4;
                                    printf("State:␣&(Q3,␣a)␣=␣Q4.\n");
                                    fprintf(write_fp, "State:␣&(Q3,␣a)␣=␣Q4.\n"
                                        );
                        }
                        else if(toupper(*p) == 'N' && state == Q4)
                        {
                                    state = Q5;
                                    printf("State:␣&(Q4,␣n)␣=␣Q5.\n");
                                    fprintf(write_fp, "State:␣&(Q4,␣n)␣=␣Q5.\n"
                                        );
                }
                else if(toupper(*p) == 'A' && state == Q5)
                {
                        state = Q6;
                        printf("State:␣&(Q5,␣a)␣=␣Q6.\n");
                        fprintf(write_fp, "State:␣&(Q5,␣a)␣=␣Q6.\n"
                            );
                }
                else if(toupper(*p) == 'R' && state == Q6)
                {
                        state = Q7;
                        printf("State:␣&(Q6,␣r)␣=␣Q7.\n");
                        fprintf(write_fp, "State:␣&(Q6,␣r)␣=␣Q7.\n"
                            );
                }
                //FINAL STATE
                if(state == Q7)
                {
                printf("|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n\n", word);
                fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n
                    \n", word);
                return true;
                }

                p++;
        }
        return false;
}

bool automFalso(char *p, FILE *write_fp)
{
        int state = Q0;
        char *word = p;

        printf("|+|+|+|AUTOMATA:␣FALSO\nWord:␣%s\n", p);
        printf("State:␣Q0\n");
        fprintf(write_fp, "|+|+|+|AUTOMATA:␣FALSO\nWord:␣%s\n", p);
        fprintf(write_fp, "State:␣Q0\n");
        while(*p != '\0')
        {
                //AUTOMATA CORE
                if(toupper(*p) == 'F' && state == Q0)
                {
                        state = Q1;
                        printf("State:␣&(Q0,␣f)␣=␣Q1.\n");
                        fprintf(write_fp, "State:␣&(Q0,␣f)␣=␣Q1.\n"
                            );
                }
                else if(toupper(*p) == 'A' && state == Q1)
                {
                        state = Q2;
                        printf("State:␣&(Q1,␣a)␣=␣Q2.\n");
                        fprintf(write_fp, "State:␣&(Q1,␣a)␣=␣Q2.\n"
                            );
                }
                else if(toupper(*p) == 'L' && state == Q2)
                {
                        state = Q3;
                        printf("State:␣&(Q2,␣l)␣=␣Q3.\n");
                        fprintf(write_fp, "State:␣&(Q2,␣l)␣=␣Q3.\n"
                            );
                }
                else if(toupper(*p) == 'S' && state == Q3)
                {
                        state = Q4;
                        printf("State:␣&(Q3,␣s)␣=␣Q4.\n");
                        fprintf(write_fp, "State:␣&(Q3,␣s)␣=␣Q4.\n"
                            );
                }
                else if(toupper(*p) == 'O' && state == Q4)
                {
                        state = Q5;
                        printf("State:␣&(Q4,␣o)␣=␣Q5.\n");
                        fprintf(write_fp, "State:␣&(Q4,␣o)␣=␣Q5.\n"
                            );
                }
                //FINAL STATE
                if(state == Q5)
                {
                printf("|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n\n", word);
                fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n
                    \n", word);
                return true;
                }

                p++;
        }
        return false;
}

bool automFalsificar(char *p, FILE *write_fp)
{
        int state = Q0;
        char *word = p;

        printf("|+|+|+|AUTOMATA:␣FALSIFICAR\nWord:␣%s\n", p);
        printf("State:␣Q0\n");
        fprintf(write_fp, "|+|+|+|AUTOMATA:␣FALSIFICAR\nWord:␣%s\n"
            , p);
        fprintf(write_fp, "State:␣Q0\n");
        while(*p != '\0')
        {
                //AUTOMATA CORE
                if(toupper(*p) == 'F' && state == Q0)
                {
                        state = Q1;
                        printf("State:␣&(Q0,␣f)␣=␣Q1.\n");
                        fprintf(write_fp, "State:␣&(Q0,␣f)␣=␣Q1.\n"
                            );
                }
                else if(toupper(*p) == 'A' && state == Q1)
                {
                        state = Q2;
                        printf("State:␣&(Q1,␣a)␣=␣Q2.\n");
                        fprintf(write_fp, "State:␣&(Q1,␣a)␣=␣Q2.\n"
                            );
                }
                else if(toupper(*p) == 'L' && state == Q2)
                {
                        state = Q3;
                        printf("State:␣&(Q2,␣l)␣=␣Q3.\n");
                        fprintf(write_fp, "State:␣&(Q2,␣l)␣=␣Q3.\n"
                            );
                }
                else if(toupper(*p) == 'S' && state == Q3)
                {
                        state = Q4;
                        printf("State:␣&(Q3,␣s)␣=␣Q4.\n");
                        fprintf(write_fp, "State:␣&(Q3,␣s)␣=␣Q4.\n"
                            );
                }
                else if(toupper(*p) == 'I' && state == Q4)
                {
                        state = Q5;
                        printf("State:␣&(Q4,␣i)␣=␣Q5.\n");
                        fprintf(write_fp, "State:␣&(Q4,␣i)␣=␣Q5.\n"
                            );
                }
                else if(toupper(*p) == 'F' && state == Q5)
                {
                        state = Q6;
                        printf("State:␣&(Q5,␣f)␣=␣Q6.\n");
                        fprintf(write_fp, "State:␣&(Q5,␣f)␣=␣Q6.\n"
                            );
                }
                else if(toupper(*p) == 'I' && state == Q6)
                {
                        state = Q7;
                        printf("State:␣&(Q6,␣i)␣=␣Q7.\n");
                        fprintf(write_fp, "State:␣&(Q6,␣i)␣=␣Q7.\n"
                            );
                }
                else if(toupper(*p) == 'C' && state == Q7)
                {
                        state = Q8;
                        printf("State:␣&(Q7,␣c)␣=␣Q8.\n");
                        fprintf(write_fp, "State:␣&(Q7,␣c)␣=␣Q8.\n"
                            );
                }
                else if(toupper(*p) == 'A' && state == Q8)
                {
                        state = Q9;
                        printf("State:␣&(Q8,␣a)␣=␣Q9.\n");
                        fprintf(write_fp, "State:␣&(Q8,␣a)␣=␣Q9.\n"
                            );
                }
                else if(toupper(*p) == 'R' && state == Q9)
                {
                        state = Q10;
                        printf("State:␣&(Q9,␣r)␣=␣Q10.\n");
```

```
                        fprintf(write_fp, "State:␣&(Q9,␣r)␣=␣Q10.\n
                            ");
                }
                //FINAL STATE
                if(state == Q10)
                {
                printf("|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n\n", word);
                fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n
                    \n", word);
                return true;
                }

                p++;
        }
        return false;
}

bool automMentir(char *p, FILE *write_fp)
{
        int state = Q0;
        char *word = p;

        printf("|+|+|+|AUTOMATA:␣MENTIR\nWord:␣%s\n", p);
        printf("State:␣Q0\n");
        fprintf(write_fp, "|+|+|+|AUTOMATA:␣MENTIR\nWord:␣%s\n", p)
            ;
        fprintf(write_fp, "State:␣Q0\n");
        while(*p != '\0')
        {
                //AUTOMATA CORE
                if(toupper(*p) == 'M' && state == Q0)
                {
                        state = Q1;
                        printf("State:␣&(Q0,␣m)␣=␣Q1.\n");
                        fprintf(write_fp, "State:␣&(Q0,␣m)␣=␣Q1.\n"
                            );
                }
                else if(toupper(*p) == 'E' && state == Q1)
                {
                        state = Q2;
                        printf("State:␣&(Q1,␣e)␣=␣Q2.\n");
                        fprintf(write_fp, "State:␣&(Q1,␣e)␣=␣Q2.\n"
                            );
                }
                else if(toupper(*p) == 'N' && state == Q2)
                {
                        state = Q3;
                        printf("State:␣&(Q2,␣n)␣=␣Q3.\n");
                        fprintf(write_fp, "State:␣&(Q2,␣n)␣=␣Q3.\n"
                            );
                }
                else if(toupper(*p) == 'T' && state == Q3)
                {
                        state = Q4;
                        printf("State:␣&(Q3,␣t)␣=␣Q4.\n");
                        fprintf(write_fp, "State:␣&(Q3,␣t)␣=␣Q4.\n"
                            );
                }
                else if(toupper(*p) == 'I' && state == Q4)
                {
                        state = Q5;
                        printf("State:␣&(Q4,␣i)␣=␣Q5.\n");
                        fprintf(write_fp, "State:␣&(Q4,␣i)␣=␣Q5.\n"
                            );
                }
                else if(toupper(*p) == 'R' && state == Q5)
                {
                        state = Q6;
                        printf("State:␣&(Q5,␣r)␣=␣Q6.\n");
                        fprintf(write_fp, "State:␣&(Q5,␣r)␣=␣Q6.\n"
                            );
                }
                //FINAL STATE
                if(state == Q6)
                {
                printf("|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n\n", word);
                fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n
                    \n", word);
                return true;
                }

                p++;
        }
        return false;
}

bool automRobar(char *p, FILE *write_fp)
{
        int state = Q0;
        char *word = p;

        printf("|+|+|+|AUTOMATA:␣ROBAR\nWord:␣%s\n", p);
        printf("State:␣Q0\n");
        fprintf(write_fp, "|+|+|+|AUTOMATA:␣ROBAR\nWord:␣%s\n", p);
        fprintf(write_fp, "State:␣Q0\n");
        while(*p != '\0')
        {
                //AUTOMATA CORE
                if(toupper(*p) == 'R' && state == Q0)
                {
                        state = Q1;
                        printf("State:␣&(Q0,␣r)␣=␣Q1.\n");
                        fprintf(write_fp, "State:␣&(Q0,␣r)␣=␣Q1.\n"
                            );
                }
                else if(toupper(*p) == 'O' && state == Q1)
                {
                        state = Q2;
                        printf("State:␣&(Q1,␣o)␣=␣Q2.\n");
                        fprintf(write_fp, "State:␣&(Q1,␣o)␣=␣Q2.\n"
                            );
                }
                else if(toupper(*p) == 'B' && state == Q2)
                {
                        state = Q3;
                        printf("State:␣&(Q2,␣b)␣=␣Q3.\n");
                        fprintf(write_fp, "State:␣&(Q2,␣b)␣=␣Q3.\n"
                            );
                }
                else if(toupper(*p) == 'A' && state == Q3)
                {
                        state = Q4;
                        printf("State:␣&(Q3,␣a)␣=␣Q4.\n");
                        fprintf(write_fp, "State:␣&(Q3,␣a)␣=␣Q4.\n"
                            );
                }
                else if(toupper(*p) == 'R' && state == Q4)
                {
                        state = Q5;
                        printf("State:␣&(Q4,␣r)␣=␣Q5.\n");
                        fprintf(write_fp, "State:␣&(Q4,␣r)␣=␣Q5.\n"
                            );
                }
                //FINAL STATE
                if(state == Q5)
                {
                printf("|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n\n", word);
                fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n
                    \n", word);
                return true;
                }

                p++;
        }
        return false;
}

bool automViolencia(char *p, FILE *write_fp)
{
        int state = Q0;
        char *word = p;

        printf("|+|+|+|AUTOMATA:␣VIOLENCIA\nWord:␣%s\n", p);
        printf("State:␣Q0\n");
        fprintf(write_fp, "|+|+|+|AUTOMATA:␣VIOLENCIA\nWord:␣%s\n",
             p);
        fprintf(write_fp, "State:␣Q0\n");
        while(*p != '\0')
        {
                //AUTOMATA CORE
                if(toupper(*p) == 'V' && state == Q0)
                {
                        state = Q1;
                        printf("State:␣&(Q0,␣v)␣=␣Q1.\n");
                        fprintf(write_fp, "State:␣&(Q0,␣v)␣=␣Q1.\n"
                            );
                }
                else if(toupper(*p) == 'I' && state == Q1)
                {
                        state = Q2;
                        printf("State:␣&(Q1,␣i)␣=␣Q2.\n");
```

```
            fprintf(write_fp, "State:␣&(Q1,␣i)␣=␣Q2.\n"
                 );
}
else if(toupper(*p) == 'O' && state == Q2)
{
            state = Q3;
            printf("State:␣&(Q2,␣o)␣=␣Q3.\n");
            fprintf(write_fp, "State:␣&(Q2,␣o)␣=␣Q3.\n"
                 );
}
else if(toupper(*p) == 'L' && state == Q3)
{
            state = Q4;
            printf("State:␣&(Q3,␣l)␣=␣Q4.\n");
            fprintf(write_fp, "State:␣&(Q3,␣l)␣=␣Q4.\n"
                 );
}
else if(toupper(*p) == 'E' && state == Q4)
{
            state = Q5;
            printf("State:␣&(Q4,␣e)␣=␣Q5.\n");
            fprintf(write_fp, "State:␣&(Q4,␣e)␣=␣Q5.\n"
                 );
}
else if(toupper(*p) == 'N' && state == Q5)
{
            state = Q6;
            printf("State:␣&(Q5,␣n)␣=␣Q6.\n");
            fprintf(write_fp, "State:␣&(Q5,␣n)␣=␣Q6.\n"
                 );
}
else if(toupper(*p) == 'C' && state == Q6)
{
            state = Q7;
            printf("State:␣&(Q6,␣c)␣=␣Q7.\n");
            fprintf(write_fp, "State:␣&(Q6,␣c)␣=␣Q7.\n"
                 );
}
else if(toupper(*p) == 'I' && state == Q7)
{
            state = Q8;
            printf("State:␣&(Q7,␣i)␣=␣Q8.\n");
            fprintf(write_fp, "State:␣&(Q7,␣i)␣=␣Q8.\n"
                 );
}
else if(toupper(*p) == 'A' && state == Q8)
{
            state = Q9;
            printf("State:␣&(Q8,␣a)␣=␣Q9.\n");
            fprintf(write_fp, "State:␣&(Q8,␣a)␣=␣Q9.\n"
                 );
}
//FINAL STATE
if(state == Q9)
{
printf("|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n\n", word);
```

```
        fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS␣WORD:␣%s\n
            \n", word);
        return true;
        }

        p++;
    }
    return false;
}

bool automTraficar(char *p, FILE *write_fp)
{
    int state = Q0;
    char *word = p;

    printf("|+|+|+|AUTOMATA:␣TRAFICAR\nWord:␣%s\n", p);
    printf("State:␣Q0\n");
    fprintf(write_fp, "|+|+|+|AUTOMATA:␣TRAFICAR\nWord:␣%s\n",
        p);
    fprintf(write_fp, "State:␣Q0\n");
    while(*p != '\0')
    {
        //AUTOMATA CORE
        if(toupper(*p) == 'T' && state == Q0)
        {
            state = Q1;
            printf("State:␣&(Q0,␣t)␣=␣Q1.\n");
            fprintf(write_fp, "State:␣&(Q0,␣t)␣=␣Q1.\n"
                );
        }
        else if(toupper(*p) == 'R' && state == Q1)
        {
            state = Q2;
            printf("State:␣&(Q1,␣r)␣=␣Q2.\n");
            fprintf(write_fp, "State:␣&(Q1,␣r)␣=␣Q2.\n"
                );
        }
        else if(toupper(*p) == 'A' && state == Q2)
        {
            state = Q3;
            printf("State:␣&(Q2,␣a)␣=␣Q3.\n");
            fprintf(write_fp, "State:␣&(Q2,␣a)␣=␣Q3.\n"
                );
        }
        else if(toupper(*p) == 'F' && state == Q3)
        {
            state = Q4;
            printf("State:␣&(Q3,␣f)␣=␣Q4.\n");
            fprintf(write_fp, "State:␣&(Q3,␣f)␣=␣Q4.\n"
                );
        }
        else if(toupper(*p) == 'I' && state == Q4)
        {
            state = Q5;
            printf("State:␣&(Q4,␣i)␣=␣Q5.\n");
            fprintf(write_fp, "State:␣&(Q4,␣i)␣=␣Q5.\n"
                );
                }
                else if(toupper(*p) == 'C' && state == Q5)
                {
                        state = Q6;
                        printf("State:␣&(Q5,␣c)␣=␣Q6.\n");
                        fprintf(write_fp, "State:␣&(Q5,␣c)␣=␣Q6.\n"
                            );
                }
                else if(toupper(*p) == 'A' && state == Q6)
                {
                        state = Q7;
                        printf("State:␣&(Q6,␣a)␣=␣Q7.\n");
                        fprintf(write_fp, "State:␣&(Q6,␣a)␣=␣Q7.\n"
                            );
                }
                else if(toupper(*p) == 'R' && state == Q7)
                {
                        state = Q8;
                        printf("State:␣&(Q7,␣r)␣=␣Q8.\n");
                        fprintf(write_fp, "State:␣&(Q7,␣r)␣=␣Q8.\n"
                            );
                }
                //FINAL STATE
                if(state == Q8)
                {
                printf("|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n\n", word);
                fprintf(write_fp, "|+|+|+|+|+|SUSPICIOUS_WORD:␣%s\n
                    \n", word);
                return true;
                }

                p++;
        }
        return false;
}
```