

Prácticas de Autómatas y Lenguajes
Curso 18-19
Grupo 1311 jueves 9-11

Objetivos de la sesión

En esta sesión nos centraremos

- Presentación de la asignatura
- Presentación de la primera práctica
- Avance en el desarrollo de la primera práctica

Presentación de la asignatura

- Leer guía y ver sobre todo la parte de evaluación de la actividad de clase

Presentación de la práctica 1

OBS. sobre la gestión de errores

- A lo largo de estas prácticas nos comprometemos a hacer un uso razonable de tus funciones por lo que debes hacer un control “mínimo” y “razonable” de gestión de errores

TAD alfabeto

1. Define una estructura C para implementar el TAD Alfabeto <ul style="list-style-type: none">• Un nombre que lo identifique (mediante un char*)• Un tamaño (número de “símbolos” que puede guardar)• Una colección de esos símbolos que tienen que poder identificarse mediante un nombre (char *)• Define un tipo de dato Alfabeto (lo ideal es que sea un struct) para el TAD que sea susceptible de utilizar punteros a él (Alfabeto *)
--

2. Implementa una función de cabecera y funcionalidad como sigue
--

```
Alfabeto * alfabetoNuevo(char * nombre, int tamano);
```

- Dimensionar si es necesario la estructura Alfabeto
- Asignar una copia en memoria nueva del nombre
- Dimensionar la colección de “símbolos” para que pueda almacenar el tamaño asignado

3. Implementa una función de cabecera y funcionalidad

```
void alfabetoElimina(Alfabeto * p_alfabeto)
```

- Libera todos los recursos asociados con el alfabeto

4. Pruebala con el siguiente programa principal

- Compila y asegúrate de que funciona correctamente respecto a la gestión de memoria

```
> valgrind --leak-check=yes ./pr_alfabeto
```

```
#include "alfabeto.h"

int main(int argc, char ** argv)
{

    Alfabeto * binario;

    binario = alfabetoNuevo("0_1",2);
    alfabetoElimina(binario);

    return 0;
}
```

5. Implementa la siguiente pareja de funciones

```
Alfabeto * alfabetoInsertaSimbolo(Alfabeto * p_alfabeto, char * simbolo);
```

- Guarda en la colección de símbolos del alfabeto una copia en memoria nueva del argumento proporcionado

```
void alfabetoImprime(FILE * fd, Alfabeto * p_alf);
```

- Imprime el alfabeto por el FILE * argumento

6. Pruébalas con el siguiente programa

```
#include <stdio.h>

#include "alfabeto.h"

int main(int argc, char ** argv)
{
    Alfabeto * binario;
    int i;

    binario = alfabetoNuevo("0_1",2);

    alfabetoInsertaSimbolo(binario,"el_0");
    alfabetoImprime(stdout, binario);

    alfabetoInsertaSimbolo(binario,"el_1");
    alfabetoImprime(stdout, binario);

    alfabetoElimina(binario);
}
```

```
    return 0;
}
```

- Comprueba que tu programa tiene una salida parecida a

```
0_1={ el_0 }
0_1={ el_0 el_1 }
```

SÓLO DE MANERA OPCIONAL

- Si has implementado la colección de símbolos del alfabeto como un vector o con otra estructura de datos C en la que sea fácil y natural acceder por “posición”

7. Implementa las siguientes dos funciones

```
char* alfabetoSmboloEn(Alfabeto * p_alf, int i);
```

- Devuelve el símbolo que está en la posición pasada como argumento en el alfabeto proporcionado como argumento.
- No se hace copia en memoria nueva, se devuelve un puntero a la cadena.

```
int alfabetoIndiceDeSmbolo(Alfabeto * p_alf, char * simbolo);
```

- Se devuelve la posición en la que está el alfabeto

8. Pruébalas con el siguiente programa principal

```
#include <stdio.h>

#include "alfabeto.h"

int main(int argc, char ** argv)
{
    Alfabeto * vocales;
    Alfabeto * binario;
    int i;

    vocales = alfabetoNuevo("vocales",5);

    alfabetoInsertaSmbolo(vocales,"la_a");
    alfabetoImprime(stdout, vocales);

    alfabetoInsertaSmbolo(vocales,"la_e");
    alfabetoImprime(stdout, vocales);

    alfabetoInsertaSmbolo(vocales,"la_i");
    alfabetoImprime(stdout, vocales);

    alfabetoInsertaSmbolo(vocales,"la_o");
    alfabetoImprime(stdout, vocales);

    alfabetoInsertaSmbolo(vocales,"la_u");
    alfabetoImprime(stdout, vocales);

    for (i = 0; i < alfabetoTamano(vocales); i++ )
    {
```

```

        fprintf(stdout,"vocales[%d]: %s\n",i,alfabetoSmboloEn(vocales,i));
    }
    for (i = 0; i < alfabetoTamano(vocales); i++ )
    {
        fprintf(stdout,"%s es vocales[%d]\n",
            alfabetoSmboloEn(vocales,i),
            alfabetoIndiceDeSmbolo(vocales,alfabetoSmboloEn(vocales,i)));
    }

    alfabetoElimina(vocales);

    binario = alfabetoNuevo("0_1",2);

    alfabetoInsertaSmbolo(binario,"0");
    alfabetoImprime(stdout, binario);

    alfabetoInsertaSmbolo(binario,"1");
    alfabetoImprime(stdout, binario);

    for (i = 0; i < alfabetoTamano(binario); i++ )
    {
        fprintf(stdout,"binario[%d]: %s\n",i,alfabetoSmboloEn(binario,i));
    }
    for (i = 0; i < alfabetoTamano(binario); i++ )
    {
        fprintf(stdout,"%s es binario[%d]\n",
            alfabetoSmboloEn(binario,i),
            alfabetoIndiceDeSmbolo(binario,alfabetoSmboloEn(binario,i)));
    }

    alfabetoElimina(binario);

    return 0;
}

```

- Comprueba que tu programa tiene una salida parecida a

```

vocales={ la_a }
vocales={ la_a la_e }
vocales={ la_a la_e la_i }
vocales={ la_a la_e la_i la_o }
vocales={ la_a la_e la_i la_o la_u }
vocales[0]: la_a
vocales[1]: la_e
vocales[2]: la_i
vocales[3]: la_o
vocales[4]: la_u
la_a es vocales[0]
la_e es vocales[1]
la_i es vocales[2]
la_o es vocales[3]
la_u es vocales[4]
0_1={ 0 }
0_1={ 0 1 }
binario[0]: 0
binario[1]: 1
0 es binario[0]
1 es binario[1]

```

TAD estado

1. Define una estructura C para implementar el TAD Estado con las siguientes características

- Un nombre que lo identifique
- Un tipo compatible con las siguientes definiciones (puedes utilizar el mecanismo que quieras pero utiliza los mismos nombres)

```
#define INICIAL 0
#define FINAL 1
#define INICIAL_Y_FINAL 2
#define NORMAL 3
```

- Define un tipo de dato Estado (lo ideal es que sea un struct) para el TAD que sea susceptible de utilizar punteros a él (Estado *)

2. Implementa una función con la siguiente cabecera

```
Estado * estadoNuevo( char * nombre, int tipo);
```

- Que reserva espacio para el Estado
- Incluyendo una copia en memoria propia del nombre que se proporciona como argumento.

3. Implementa una función con la siguiente cabecera

```
void estadoElimina( Estado * p_s);
```

- Que elimina todos los recursos asociados con el estado que se proporciona como argumento

4. Pruébalo con el siguiente programa principal

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "estado.h"

int main (int argc, char ** argv)
{
    Estado * estados [10];
    int i;
    char nombre_estado[20];

    for (i=0; i< 10; i++)
    {

        sprintf(nombre_estado,"s_%d",i);
        switch ( i % 4 )
```

```

        {
            case 0:
                estados[i] =
                    estadoNuevo(nombre_estado, INICIAL_Y_FINAL);
                break;
            case 1:
                estados[i] = estadoNuevo(nombre_estado, INICIAL);
                break;
            case 2:
                estados[i] = estadoNuevo(nombre_estado, FINAL);
                break;
            case 3:
                estados[i] = estadoNuevo(nombre_estado, NORMAL);
                break;
        }
    }

    for (i=0; i< 10; i++)
    {
        estadoElimina(estados[i]);
    }

    return 0;
}

```

5. Implementa una función de cabecera

`void estadoImprime(FILE * fd, Estado * p_s);`

- Que muestre por el FILE *fd el estado proporcionado como argumento

6. Pruébalo con el siguiente programa

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "estado.h"

int main (int argc, char ** argv)
{
    Estado * estados [10];
    int i;
    char nombre_estado[20];

    for (i=0; i< 10; i++)
    {

        sprintf(nombre_estado, "s_%d", i);
        switch ( i % 4 )
        {
            case 0:
                estados[i] =
                    estadoNuevo(nombre_estado, INICIAL_Y_FINAL);
                break;
            case 1:
                estados[i] = estadoNuevo(nombre_estado, INICIAL);
                break;

```

```

        case 2:
            estados[i] = estadoNuevo(nombre_estado, FINAL);
            break;
        case 3:
            estados[i] = estadoNuevo(nombre_estado, NORMAL);
            break;
    }

    }

    for (i=0; i< 10; i++)
    {
        estadoImprime(stdout, estados[i]);
    }

    for (i=0; i< 10; i++)
    {
        estadoElimina(estados[i]);
    }

    return 0;
}

```

7. Comprueba que la salida es lo más parecida posible a la siguiente

->s_0* ->s_1 s_2* s_3 ->s_4* ->s_5 s_6* s_7 ->s_8* ->s_9

8. Implementa las siguientes funciones

```
int estadoEs(Estado * p_s, char * nombre);
```

- Que devuelve 1 si el estado argumento tiene el nombre argumento

```
char * estadoNombre(Estado * p_s);
```

- Que devuelve un puntero (no una copia en memoria nueva) del nombre del estado argumento

```
int estadoTipo(Estado * p_s);
```

- Que devuelve el tipo del estado argumento

8. Pruébalas con el siguiente programa principal

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "estado.h"

int posEstado (Estado ** estados, char * nombre, int max)
{
    int i;

    for (i=0; i < max; i++)
    {
        if ( estadoEs(estados[i], nombre) )
        {
            return i;
        }
    }
}

```

```

        }
    }
    return -1;
}

int main (int argc, char ** argv)
{
    Estado * estados [10];
    int i;
    char nombre_estado[20];

    for (i=0; i< 10; i++)
    {

        sprintf(nombre_estado,"s_%d",i);
        switch ( i % 4 )
        {
            case 0:
                estados[i] =
                    estadoNuevo(nombre_estado,INICIAL_Y_FINAL);
                break;
            case 1:
                estados[i] = estadoNuevo(nombre_estado,INICIAL);
                break;
            case 2:
                estados[i] = estadoNuevo(nombre_estado,FINAL);
                break;
            case 3:
                estados[i] = estadoNuevo(nombre_estado,NORMAL);
                break;

        }

    }

    fprintf(stdout,"El estado %s está en la posicion %d\n", argv[1],
posEstado(estados, argv[1], 10 ) );

    for (i=0; i< 10; i++)
    {
        fprintf(stdout,"El tipo del estado %s es %d\n",
estadoNombre(estados[i]),estadoTipo(estados[i]));
    }

    for (i=0; i< 10; i++)
    {
        estadoImprime(stdout,estados[i]);
    }

    for (i=0; i< 10; i++)
    {
        estadoElimina(estados[i]);
    }

    return 0;
}

```

Para ello ejecútalo con el siguiente comando


```
> valgrind --leak-check=yes ./pr_estado_guia3 e
```

9. Comprueba que la salida es lo más parecida posible a la siguiente

```
El estado e está en la posición -1
El tipo del estado s_0 es 2
El tipo del estado s_1 es 0
El tipo del estado s_2 es 1
El tipo del estado s_3 es 3
El tipo del estado s_4 es 2
El tipo del estado s_5 es 0
El tipo del estado s_6 es 1
El tipo del estado s_7 es 3
El tipo del estado s_8 es 2
El tipo del estado s_9 es 0
->s_0* ->s_1 s_2* s_3 ->s_4* ->s_5 s_6* s_7 ->s_8* ->s_9
```

TAD palabra

1. Define una estructura C para implementar el TAD Palabra con las siguientes características

- Su tamaño (el número de letras)
- Una colección de letras representadas mediante un char * (cada una de ellas)
- Define un tipo de dato Palabra (lo ideal es que sea un struct) para el TAD que sea susceptible de utilizar punteros a él (Palabra *)

2. Implementa la pareja de funciones

```
Palabra * palabraNueva();
```

- Que reserva memoria para una Palabra nueva de tamaño 0

```
void palabraElimina(Palabra * p_p);
```

- Que libera todos los recursos de la Palabra incluyendo
 - La colección completa de letras
 - La propia estructura de la Palabra

3. Pruébalo con el siguiente programa principal

```
#include "palabra.h"

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main (int argc, char **argv)
{
    Palabra * pal;
```

```

        pal = palabraNueva();

        palabraElimina(pal);

        return 0;
}

```

4. Implementa las funciones

```
Palabra * palabraInsertaLetra(Palabra * p_p, char * letra);
```

- Reserva memoria nueva para realizar una copia de la letra argumento
- Redimensiona si es necesario la colección de letras de la Palabra argumento
- Incluye la nueva letra al principio de la palabra

```
void palabraImprime(FILE * fd, Palabra * p_p);
```

- Muestra por el FILE * argumento la Palabra

5. Pruébala con el siguiente programa principal

```

#include "palabra.h"

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main (int argc, char **argv)
{
    Palabra * pal;
    int i;
    char letra [20];
    char * letrai;

    pal = palabraNueva();

    for (i=0; i < argc-1; i++)
    {
        sprintf(letra,"l_%d%s",i,argv[1+i]);
        palabraInsertaLetra(pal,letra);
        fprintf(stdout,"pal_%d:\n",i);
        palabraImprime(stdout,pal);
        fprintf(stdout,"\n");
    }

    palabraElimina(pal);

    return 0;
}

```

- Para ello ejecútalo con el siguiente comando

```
> ./pr_palabra_guia2 a e i o u
```

6. Comprueba que la salida es lo más parecida posible a la siguiente

```

pal_0:
[ (1) l_0_a]

```

```

pal_1:
[(2) l_0_a l_1_e]

pal_2:
[(3) l_0_a l_1_e l_2_i]

pal_3:
[(4) l_0_a l_1_e l_2_i l_3_o]

pal_4:
[(5) l_0_a l_1_e l_2_i l_3_o l_4_u]

```

7. Implementa las siguientes funciones

```

char * palabraQuitaInicio(Palabra * p_p);
    • Devuelve la primera letra de la palabra
    • Eliminándola de ella

int palabraTamano(Palabra * p_p);
    • Devuelve el número de letras de la palabra

```

8. Pruébala con el siguiente programa principal

```

#include "palabra.h"

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main (int argc, char **argv)
{
    Palabra * pal;
    int i;
    char letra [20];
    char * letrai;

    pal = palabraNueva();

    for (i=0; i < argc-1; i++)
    {
        sprintf(letra,"l_%d%s",i,argv[1+i]);
        palabraInsertaLetra(pal,letra);
        fprintf(stdout,"pal %d:\n",i);
        palabraImprime(stdout,pal);
        fprintf(stdout,"\n");
    }
    while ( palabraTamano(pal) > 0 )
    {
        fprintf(stdout,
"QUITAMOS %s DE LA PALABRA QUE QUEDA ASI:\n",letrai=palabraQuitaInicio(pal));
        palabraImprime(stdout,pal);
        free(letrai);
    }

    palabraElimina(pal);

    return 0;
}

```

```
}
```

- Para ello ejecútalo con el siguiente comando

```
> ./pr_palabra_guia2 a b c d e f g h i j k l m n o p q r s t u v w x y z
```

9. Comprueba que la salida es lo más parecida posible a la siguiente

```
pal_0:
[(1) l_0_a]

pal_1:
[(2) l_0_a l_1_b]

pal_2:
[(3) l_0_a l_1_b l_2_c]

pal_3:
[(4) l_0_a l_1_b l_2_c l_3_d]

pal_4:
[(5) l_0_a l_1_b l_2_c l_3_d l_4_e]

pal_5:
[(6) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f]

pal_6:
[(7) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g]

pal_7:
[(8) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h]

pal_8:
[(9) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i]

pal_9:
[(10) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j]

pal_10:
[(11) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k]

pal_11:
[(12) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l]

pal_12:
[(13) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l l_12_m]

pal_13:
[(14) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l l_12_m
l_13_n]

pal_14:
[(15) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l l_12_m
l_13_n l_14_o]

pal_15:
[(16) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l l_12_m
l_13_n l_14_o l_15_p]

pal_16:
[(17) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l l_12_m
l_13_n l_14_o l_15_p l_16_q]

pal_17:
[(18) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l l_12_m
l_13_n l_14_o l_15_p l_16_q l_17_r]
```

[(19) 1_0_a 1_1_b 1_2_c 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m
1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s]

[(20) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l l_12_m
l_13_n l_14_o l_15_p l_16_q l_17_r l_18_s l_19_t]

[(21) l_0_a l_1_b l_2_c l_3_d l_4_e l_5_f l_6_g l_7_h l_8_i l_9_j l_10_k l_11_l l_12_m
l_13_n l_14_o l_15_p l_16_q l_17_r l_18_s l_19_t l_20_u]

[(22) 1_0_a 1_1_b 1_2_c 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m
1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v]

[(23) 1_0_a 1_1_b 1_2_c 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m
1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w]

[(24) 1_0_a 1_1_b 1_2_c 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m
1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x]

[(25) 1_0_a 1_1_b 1_2_c 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m
1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y]

```
[ (26) 1_0_a 1_1_b 1_2_c 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m
1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y
1_25_z ]
```

[(25) 1_1_b 1_2_c 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m 1_13_n
1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y 1_25_z]

[(24) 1_2_c 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m 1_13_n 1_14_o
1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y 1_25_z]

[(23) 1_3_d 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m 1_13_n 1_14_o
1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y 1_25_z]

[(22) 1_4_e 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m 1_13_n 1_14_o 1_15_p
1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y 1_25_z]

[(21) 1_5_f 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m 1_13_n 1_14_o 1_15_p 1_16_q
1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y 1_25_z]

[(20) 1_6_g 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m 1_13_n 1_14_o 1_15_p 1_16_q 1_17_r
1_18_s 1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y 1_25_z]

```
[ (19) 1_7_h 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m 1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s
1_19_t 1_20_u 1_21_v 1_22_w 1_23_x 1_24_y 1_25_z ]
```

[(18) 1_8_i 1_9_j 1_10_k 1_11_l 1_12_m 1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t
1_20_u 1_21_v 1_22_w 1_23_x 1_24_y 1_25_z]

[(17) l_9_j l_10_k l_11_l l_12_m l_13_n l_14_o l_15_p l_16_q l_17_r l_18_s l_19_t l_20_u
l_21_v l_22_w l_23_x l_24_y l_25_z]

[(16) 1_10_k 1_11_l 1_12_m 1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u
1_21_v 1_22_w 1_23_x 1_24_y 1_25_z]

$$\begin{bmatrix} (15) & \bar{1} & \bar{11} & \bar{1} & \bar{12} & \bar{m} & \bar{1} & \bar{13} & \bar{n} & \bar{1} & \bar{14} & \bar{o} & \bar{1} & \bar{15} & \bar{p} & \bar{1} & \bar{16} & \bar{q} & \bar{1} & \bar{17} & \bar{r} & \bar{1} & \bar{18} & \bar{s} & \bar{1} & \bar{19} & \bar{t} & \bar{1} & \bar{20} & \bar{u} & \bar{1} & \bar{21} & \bar{v} \\ \bar{1} & \bar{22} & \bar{w} & \bar{1} & \bar{23} & \bar{x} & \bar{1} & \bar{24} & \bar{y} & \bar{1} & \bar{25} & \bar{z} \end{bmatrix}$$

[(14) 1_12_m 1_13_n 1_14_o 1_15_p 1_16_q 1_17_r 1_18_s 1_19_t 1_20_u 1_21_v 1_22_w
1_23_x 1_24_y 1_25_z]

```
QUITAMOS l_12_m DE LA PALABRA QUE QUEDA ASI:
[(13) l_13_n l_14_o l_15_p l_16_q l_17_r l_18_s l_19_t l_20_u l_21_v l_22_w l_23_x
l_24_y l_25_z]
QUITAMOS l_13_n DE LA PALABRA QUE QUEDA ASI:
[(12) l_14_o l_15_p l_16_q l_17_r l_18_s l_19_t l_20_u l_21_v l_22_w l_23_x l_24_y
l_25_z]
QUITAMOS l_14_o DE LA PALABRA QUE QUEDA ASI:
[(11) l_15_p l_16_q l_17_r l_18_s l_19_t l_20_u l_21_v l_22_w l_23_x l_24_y l_25_z]
QUITAMOS l_15_p DE LA PALABRA QUE QUEDA ASI:
[(10) l_16_q l_17_r l_18_s l_19_t l_20_u l_21_v l_22_w l_23_x l_24_y l_25_z]
QUITAMOS l_16_q DE LA PALABRA QUE QUEDA ASI:
[(9) l_17_r l_18_s l_19_t l_20_u l_21_v l_22_w l_23_x l_24_y l_25_z]
QUITAMOS l_17_r DE LA PALABRA QUE QUEDA ASI:
[(8) l_18_s l_19_t l_20_u l_21_v l_22_w l_23_x l_24_y l_25_z]
QUITAMOS l_18_s DE LA PALABRA QUE QUEDA ASI:
[(7) l_19_t l_20_u l_21_v l_22_w l_23_x l_24_y l_25_z]
QUITAMOS l_19_t DE LA PALABRA QUE QUEDA ASI:
[(6) l_20_u l_21_v l_22_w l_23_x l_24_y l_25_z]
QUITAMOS l_20_u DE LA PALABRA QUE QUEDA ASI:
[(5) l_21_v l_22_w l_23_x l_24_y l_25_z]
QUITAMOS l_21_v DE LA PALABRA QUE QUEDA ASI:
[(4) l_22_w l_23_x l_24_y l_25_z]
QUITAMOS l_22_w DE LA PALABRA QUE QUEDA ASI:
[(3) l_23_x l_24_y l_25_z]
QUITAMOS l_23_x DE LA PALABRA QUE QUEDA ASI:
[(2) l_24_y l_25_z]
QUITAMOS l_24_y DE LA PALABRA QUE QUEDA ASI:
[(1) l_25_z]
QUITAMOS l_25_z DE LA PALABRA QUE QUEDA ASI:
[(0)]
```