Instituto Politécnico Nacional
Escuela Superior de Cómputo

Teoría de la Computación

Práctica 2. Operaciones entre lenguajes

Juárez Martínez Yunuen

4CM2

Introducción

Definiciones

Lenguaje: Es un conjunto de palabras (cadenas) de un determinado alfabeto Σ.

$$L \subset W(\Sigma)$$

Lenguaje vacío: El conjunto vacío ∅ es un subconjunto de W(Σ).

Para distinguirlos, hay que fijarse en su cardinalidad (número de símbolos).

$$C(\emptyset) = 0$$

$$C(\{\lambda\}) = 1$$

Alfabeto: Uno de los lenguajes generados por él mismo: el que contiene todas las palabras de una sola letra.

Operaciones entre Lenguajes

Unión o alternativa de lenguajes

$L1 \subset W(\sum)$

$L2 \subset W(\sum)$

$$L1 \cup L2 = \{ x \mid x \in L1 \vee x \in L2 \}$$

Propiedades

- Operación cerrada: la unión de dos lenguajes sobre el mismo alfabeto es también un lenguaje sobre dicho alfabeto.
- Propiedad asociativa: $(L1 \cup L2) \cup L3 = L1 \cup (L2 \cup L3)$
- Existencia de un elemento neutro: cualquiera que sea el lenguaje L, el lenguaje vacío $\emptyset$ cumple que $\emptyset \cup L = L \cup \emptyset = L$
- Propiedad conmutativa: cualesquiera que sean L1 y L2, se verifica que $L1 \cup L2 = L2 \cup L1$
- Propiedad idempotente: cualquiera que sea L, se verifica que $L \cup L$

Intersección de lenguajes

$L1 \subset W(\sum)$

$L2 \subset W(\sum)$

$$L1 \cap L2 = \{ x \mid x \in L1 \wedge x \in L2 \}$$

Concatenación

$L1 \subset W(\sum)$

$L2 \subset W(\sum)$

$$L1L2 = \{ x1x2 \mid x1 \in L1 \wedge x2 \in L2 \}$$

- La definición anterior sólo es válida si L1 y L2 contienen al menos un elemento.
- Para la concatenación de L con el lenguaje vacío $\emptyset$ se tiene que: $\emptyset L = L\emptyset = \emptyset$

Propiedades

- Operación cerrada: la concatenación de dos lenguajes sobre el mismo alfabeto es también un lenguaje sobre el mismo alfabeto.
- Propiedad asociativa: $(L1L2)L3 = L1(L2L3)$
- Existencia de un elemento neutro: cualquiera que sea el lenguaje L, el lenguaje de la palabra vacía cumple que: $\lambda\{L\} = L\{\lambda\} = L$

Potencia

L^i = LLL … L (i veces)

Definiremos también:

- L1 = L
- L^(i+1) = L^iL = LL^i (i > 0)
- $L^iL^j = L^{i+j}$ (i, j > 0)
- L^0 = {λ}

Cierre o clausura positiva

L+ = {L} ∪ {LL} ∪ {LLL} ∪ ⋯ = $\bigcup_{n=1}^{\infty} L^n$

- Ninguna clausura positiva contiene la palabra vacía, a menos que dicha palabra este en L.
- Puesto que el alfabeto Σ es también un lenguaje sobre Σ, puede aplicársele esta operación.

$$\Sigma+ = W(\Sigma) - \{\lambda\}$$

Cierre u operación estrella (Cerradura de Kleene)

L+ = {λ} ∪ {L} ∪ {LL} ∪ {LLL} ∪ ⋯ = $\bigcup_{n=1}^{\infty} L^n$

- Puesto que el alfabeto Σ es también un lenguaje sobre Σ, puede aplicársele esta operación.

$$\Sigma^* = W(\Sigma)$$

Identidades de cierres

- L* = L+ ∪ {λ}
- L+= L L* =L*L

Reflexión de lenguajes

L−1 = { x−1 | x ∈ L }

Es decir, es el lenguaje que contiene todas las palabras inversas de L.

Desarrollo

#include <errno.h> ← Se utiliza para identificar el tipo de error que se está presentando en cada función

#include <math.h> ← Necesaria para abs() si se trabaja con potencias.

```
// Lista enlazada
typedef struct nodo {
   char* dato;
   struct nodo* siguiente;
} Nodo;

// Estructura que representa la lista de listas (Lenguaje)
typedef struct Lista {
   Nodo* cabeza;
} Lista;

#define LISTAS 20 ← Podemos cambiar el número de cadenas que queramos agregar a la lista
Lista* arrayList[LISTAS];
int numListas = 0;
```

Funciones

void liberarLista(Nodo* cabeza); ← Libera una lista para que la podamos ocupar o sobrescribir.

void mostrarListasTodas(); ← Muestra nuestras listas para que podamos seguir escribiendo.

int agregarListaNueva(); ← Agrega un valor al final de la lista en el índice dado, verificando duplicados para no repetir palabras.

int agregaraLista(int indice, const char* valor); ← Agrega las operaciones realizadas a una lista nueva dentro de la lista de listas.

Nodo* crearNodo(const char* valor); ← Crea un nuevo nodo, asignando memoria para el nodo y para la cadena.

void guardarArchivo(Lista* lista, const char* nombre); ← Para guardar los archivos generados de cada operación.

int existeEnLista(Nodo* cabeza, const char* dato); ← Verifica si un dato ya existe en la lista.

void invertirCadena(char* cad); ← Invierte la cadena.

void mostrarLista(Nodo* cabeza); ← Muestra los elementos de una lista.

Lista* copiarLista(Lista* origen); ← Copia una lista a una lista nueva.

char* potenciarCadena(const char* base, int n); ← Hace la potencia de una cadena infinitamente.

```
void menu(int indice);
void unioN(int indice);
void concatenacion(int indice);
void potencia(int indice);
void cerraduraPositiva(int indice);
```

```
void cerraduraKleane(int indice);
void reflexion(int indice);
void limpiarBuffer();

// Crea un nuevo nodo, asignando memoria para el nodo y para la cadena
Nodo* crearNodo(const char* valor) {
    Nodo* nuevo = (Nodo*)malloc(sizeof(Nodo));
    if (nuevo == NULL) {
        perror("Error: memoria insuficiente para Nodo");
        return NULL;
    }

    nuevo->dato = strdup(valor);
    if (nuevo->dato == NULL) {
        perror("Error: asignacion de memoria para dato");
        free(nuevo);
        return NULL;
    }
    nuevo->siguiente = NULL;
    return nuevo;
}

void liberarNodo(Nodo* nodo) {
    if (nodo != NULL) {
        free(nodo->dato);
        free(nodo);
    }
}

void liberarLista(Nodo* cabeza) {
    Nodo* actual = cabeza;
    while (actual != NULL) {
        Nodo* temp = actual;
        actual = actual->siguiente;
        liberarNodo(temp);
```

```
    }
}

void inicializarArreglo() {
    for (int i = 0; i < LISTAS; i++) {
        arrayList[i] = (Lista*)malloc(sizeof(Lista));
        if (arrayList[i] == NULL) {
            fprintf(stderr, "Error al asignar memoria para Lista %d\n", i);
            exit(EXIT_FAILURE);
        }
        arrayList[i]->cabeza = NULL;
    }
    numListas = 0;
}

int agregarListaNueva() {
    if (numListas < LISTAS) {
        int i = numListas;
        // solo asegura que esté vacía
        if (arrayList[i]->cabeza != NULL) {
            liberarLista(arrayList[i]->cabeza);
            arrayList[i]->cabeza = NULL;
        }
        numListas++;
        printf("Lista agregada: L%d inicializada.\n", i);
        return i;
    } else {
        printf("Error: El arreglo de listas esta lleno. Maximo %d listas.\n", LISTAS);
        return -1;
    }
}

void mostrarListasTodas() {
    printf("\n\t Listas disponibles: \n");
```

```
    for (int i = 0; i < numListas; i++) {
        Nodo* actual = arrayList[i]->cabeza;
        printf("L%d: {", i);

        if (actual == NULL) {
            printf("λ");
        } else {
            while (actual != NULL) {
                printf("%s", actual->dato);
                actual = actual->siguiente;
                if (actual != NULL) {
                    printf(", ");
                }
            }
        }
        printf("}\n");
    }
    printf("\n");
}

void liberarMemoria() {
    for (int i = 0; i < LISTAS; i++) {

        liberarLista(arrayList[i]->cabeza);
        arrayList[i]->cabeza = NULL;
        free(arrayList[i]);
        arrayList[i] = NULL;
    }
    numListas = 0;
    printf("\nMemoria liberada.\n");
}

// Agrega un valor al final de la lista en el índice dado, verificando duplicados para no repetir palabras
int agregaraLista(int indice, const char* valor) {
    if (indice < 0 || indice >= numListas || arrayList[indice] == NULL) {
```

```
        printf("Indice de lista fuera de rango o no inicializado.\n");
        return 0;
    }


    if (existeEnLista(arrayList[indice]->cabeza, valor)) {
        //printf("Advertencia: '%s' ya existe en L%d. No se agrega.\n", valor, indice);
        return 1;
    }

    Nodo* nuevo = crearNodo(valor);
    if (nuevo == NULL) {
        return 0;
    }

    if (arrayList[indice]->cabeza == NULL) {
        arrayList[indice]->cabeza = nuevo;
    } else {
        Nodo* actual = arrayList[indice]->cabeza;
        while (actual->siguiente != NULL) {
            actual = actual->siguiente;
        }
        actual->siguiente = nuevo;
    }
    return 1;
}

// Muestra los elementos de una lista
void mostrarLista(Nodo* cabeza) {
    Nodo* actual = cabeza;
    printf("Resultado: {");

    if (actual == NULL) {
        printf("λ");
    } else {
```

```
        while (actual != NULL) {
            printf("%s", actual->dato);
            actual = actual->siguiente;
            if (actual != NULL) {
                printf(", ");
            }
        }
    }
    printf("}\n");
}

// Verifica si un dato ya existe en la lista
int existeEnLista(Nodo* cabeza, const char* dato) {
    Nodo* actual = cabeza;
    while (actual != NULL) {
        if (strcmp(actual->dato, dato) == 0) {
            return 1; // Existe
        }
        actual = actual->siguiente;
    }
    return 0; // No existe
}

// Invierte la cadena
void invertirCadena(char* cad) {
    int lon = strlen(cad);
    char temp;

    for (int i = 0, j = lon - 1; i < j; i++, j--) {
        temp = cad[i];
        cad[i] = cad[j];
        cad[j] = temp;
    }
}
```

```
// Copia una lista a una lista nueva
Lista* copiarLista(Lista* origen) {
    Lista* nueva = (Lista*)malloc(sizeof(Lista));
    if (nueva == NULL) {
        perror("Error al asignar memoria para la nueva lista");
        return NULL;
    }
    nueva->cabeza = NULL;
    Nodo* actual = origen->cabeza;

    while (actual != NULL) {
        // Usar agregaraLista para añadir el dato a la nueva lista
        agregaraLista(numListas - 1, actual->dato);
        actual = actual->siguiente;
    }
    return nueva;

    Lista* copia = (Lista*)malloc(sizeof(Lista));
    if (copia == NULL) {
        perror("Error al asignar memoria para copia de lista");
        return NULL;
    }
    copia->cabeza = NULL;
    Nodo** p_siguiente = &copia->cabeza;
    Nodo* actual_origen = origen->cabeza;

    while (actual_origen != NULL) {
        Nodo* nuevo_nodo = crearNodo(actual_origen->dato);
        if (nuevo_nodo == NULL) {
            // Manejar error y liberar memoria de lo copiado hasta ahora
            liberarLista(copia->cabeza);
            free(copia);
            return NULL;
        }
        *p_siguiente = nuevo_nodo;
```

```
        p_siguiente = &nuevo_nodo->siguiente;
        actual_origen = actual_origen->siguiente;
    }
    return copia;
}

//Para los archivos

void guardarArchivo(Lista* lista, const char* nombre) {
    FILE* arch;
    arch = fopen(nombre, "w");

    if (arch == NULL) {
        perror("Error al abrir archivo para guardar");
        return;
    }

    Nodo* actual = lista->cabeza;

    // Escribir los elementos de la lista separados por espacio
    while (actual != NULL) {
        fprintf(arch, "%s ", actual->dato);
        actual = actual->siguiente;
    }

    fclose(arch);
    printf("Lista guardada en el archivo: %s\n", nombre);
}

char* informacionArchivo(const char* nombreArch) {
    FILE *fp = NULL;
    char *buffer = NULL;
    long tam = 0;

    fp = fopen(nombreArch, "r");
```

```
    if (fp == NULL) {
        fprintf(stderr, "Error al abrir el archivo: %s\n", strerror(errno));
        return NULL;
    }

    if (fseek(fp, 0, SEEK_END) != 0) {
        fprintf(stderr, "Error al buscar el final del archivo %s\n", strerror(errno));
        fclose(fp);
        return NULL;
    }

    tam = ftell(fp);

    if (tam == -1) {
        fprintf(stderr, "Error al obtener el tamanio del archivo %s\n", strerror(errno));
        fclose(fp);
        return NULL;
    }

    rewind(fp);

    buffer = (char*)malloc(tam + 1);

    if (buffer == NULL) {
        fprintf(stderr, "Error al asignar memoria para el archivo %s\n", strerror(errno));
        fclose(fp);
        return NULL;
    }
    // Para archivos de texto, es más seguro leer por líneas o usar un bucle con fgetc. Es importante que el archivo no contenga el byte '\0' dentro.
    if (fread(buffer, tam, 1, fp) != 1) {
        if (ferror(fp)) {
            fprintf(stderr, "Error al leer el contenido del archivo %s\n", strerror(errno));
        } else {
        }
```

```
        free(buffer);
        fclose(fp);
        return NULL;
    }

    buffer[tam] = '\0';

    fclose(fp);
    return buffer;
}

void procesarPalabras(int indice, char* cadena) {
    if (cadena == NULL) {
        printf("Cadena nula recibida para el indice %d.\n", indice);
        return;
    }

    char delimitador[] = " \n\r\t";
    char* token;

    token = strtok(cadena, delimitador);

    while (token != NULL) {
        if (strlen(token) > 0) {
            agregaraLista(indice, token);
        }
        token = strtok(NULL, delimitador);
    }
}
```

Funciones de Operaciones con Lenguajes

/* Está función crea una lista nueva dentro de la lista de listas y agrega con la que el usuario decide en el main o en el menu (dependiendo de la vez que

quiera ejecutar la funcion) y pregunta con cual quiere hacer la función de union para añadir las partes no existentes a esta nueva lista junto con las que

ya había agregado y al final guarda unicamente la lista creada en un archivo */

```
void unioN(int indiceL1) {
    int indiceL2, indiceResultado;

    printf("\n--- Operacion: Union ---\n");
    mostrarListasTodas();

    printf("El lenguaje L%d sera el primero. \n", indiceL1);
    printf("Ingresa el indice del segundo lenguaje (L): ");

    if (scanf("L%d", &indiceL2) != 1) {
        printf("Entrada invalida. Operacion cancelada.\n");
        while(getchar() != '\n');
        return;
    }

    if (indiceL2 < 0 || indiceL2 >= numListas || indiceL2 == indiceL1) {
        printf("Indice de lista no valido o igual al lenguaje actual.\n");
        return;
    }

    indiceResultado = agregarListaNueva();
    if (indiceResultado == -1) return;

    Lista* L1 = arrayList[indiceL1];
    Lista* L2 = arrayList[indiceL2];
    Lista* LResultado = arrayList[indiceResultado];

    //Copia L1 a LResultado
    Nodo* actual = L1->cabeza;
    while (actual != NULL) {
        agregaraLista(indiceResultado, actual->dato);
        actual = actual->siguiente;
    }

    actual = L2->cabeza;
```

```
    while (actual != NULL) {
        agregaraLista(indiceResultado, actual->dato);
        actual = actual->siguiente;
    }

    printf("Resultado de L%d U L%d generado en L%d.\n", indiceL1, indiceL2, indiceResultado);
    mostrarLista(LResultado->cabeza);
    guardarArchivo(LResultado, "ResultadoUnion.txt");
}

/* Esta funcion agrega a una lista nueva (en la lista de listas) cada elemento de la lista elegida por el usuario en el main o menu a cada elemento de la segunda lista elegida:
    L1 = {gato, perro, caballo}
    L2 = {animal, comida}
    LR = {gatoanimal, gatocomida, perroanimal, perrocomida, caballoanimal, caballocomida} (Nota: Siguiendo la definición estándar L1.L2)
Y la cadena resultante se guarda en un archivo */
void concatenacion(int indiceL1) {
    int indiceL2, indiceResultado;

    printf("\n--- Operacion: Concatenación ---\n");
    mostrarListasTodas();

    printf("El lenguaje L%d sera el primero. \n", indiceL1);
    printf("Ingresa el indice del segundo lenguaje (L): ");

    if (scanf("L%d", &indiceL2) != 1) {
        printf("Entrada invalida. Operacion cancelada.\n");
        while(getchar() != '\n');
        return;
    }

    if (indiceL2 < 0 || indiceL2 >= numListas) {
        printf("Indice de lista no valido.\n");
        return;
    }
```

```
indiceResultado = agregarListaNueva();
if (indiceResultado == -1) return;

Lista* L1 = arrayList[indiceL1];
Lista* L2 = arrayList[indiceL2];
Lista* LResultado = arrayList[indiceResultado];

Nodo* nodoL1 = L1->cabeza;

// Caso especial para cadena vacía (λ): Si L1 o L2 son {λ}, la concatenación es el otro conjunto.
int L1_vacia = (L1->cabeza == NULL || strcmp(L1->cabeza->dato, "λ") == 0);
int L2_vacia = (L2->cabeza == NULL || strcmp(L2->cabeza->dato, "λ") == 0);

if (L1->cabeza == NULL && L2->cabeza == NULL) {
   // Concatenación de dos lenguajes vacíos es un lenguaje vacío
   // No hacer nada
} else if (L1->cabeza == NULL) {
   // {} . L2 = {}
} else if (L2->cabeza == NULL) {
   // L1 . {} = {}
} else {
   Nodo* actualL1 = L1->cabeza;
   while (actualL1 != NULL) {
      Nodo* actualL2 = L2->cabeza;
      while (actualL2 != NULL) {

         int len1 = strlen(actualL1->dato);
         int len2 = strlen(actualL2->dato);

         char* nuevaCadena = (char*)malloc(len1 + len2 + 1);
         if (nuevaCadena == NULL) {
            perror("Error de memoria en concatenacion");
            return;
         }
```

```
            strcpy(nuevaCadena, actualL1->dato);
            strcat(nuevaCadena, actualL2->dato);

            agregaraLista(indiceResultado, nuevaCadena);

            free(nuevaCadena);
            actualL2 = actualL2->siguiente;
        }
        actualL1 = actualL1->siguiente;
    }
  }

  printf("Resultado de L%d . L%d generado en L%d.\n", indiceL1, indiceL2, indiceResultado);
  mostrarLista(LResultado->cabeza);
  guardarArchivo(LResultado, "ResultadoConcatenacion.txt");
}

char* potenciarCadena(const char* base, int n) {
  if (n <= 0) {
    return strdup("");
  }

  int base_len = strlen(base);
  // Calcular el tamaño total + 1 para el '\0'
  long total_len = (long)base_len * n + 1;

  if (total_len > 1000000) { // Límite razonable para evitar fallos de memoria
    fprintf(stderr, "Error: Cadena de potencia demasiado larga.\n");
    return strdup("");
  }

  char* resultado = (char*)malloc(total_len);
  if (resultado == NULL) {
    perror("Error de asignacion de memoria para potenciarCadena");
```

```
        return strdup("");
    }

    resultado[0] = '\0';

    for (int i = 0; i < n; i++) {
        strcat(resultado, base);
    }

    return resultado;
}

/* Se crea una lista nueva (en la lista de listas) donde se agrega la potencia que pide el usuario de la cadena, es decir,
la última potencia
Nota: El rango de las potencias posibles será de -5 a 10 */
void potencia(int indiceL1) {
    int pot, indiceResultado;

    printf("\n--- Operacion: Potencia (L^n) ---\n");
    mostrarListasTodas();
    printf("Lenguaje a potenciar: L%d\n", indiceL1);

    printf("Ingresa el valor de la potencia (rango: -5 a 10): ");

    if (scanf("%d", &pot) != 1) {
        printf("Entrada invalida. Operacion cancelada.\n");
        limpiarBuffer();
        return;
    }
    limpiarBuffer();

    if (pot < -5 || pot > 10) {
        printf("Potencia fuera de rango (-5 a 10).\n");
        return;
```

```
}

Lista* L1 = arrayList[indiceL1];

indiceResultado = agregarListaNueva();
if (indiceResultado == -1) return;
Lista* LResultado = arrayList[indiceResultado];

if (pot == 0) {
   agregaraLista(indiceResultado, "λ");
} else {

   int absPot = abs(pot);

   Nodo* actual = L1->cabeza;
   while (actual != NULL) {
      char* cadBase = actual->dato;
      char* cadInvertida = NULL;
      char* cadFinal = NULL;

      if (pot < 0) {
         cadInvertida = strdup(cadBase);
         if (cadInvertida == NULL) {
            perror("Error de memoria en reflexion (potencia negativa)");
            return;
         }
         invertirCadena(cadInvertida);
         cadFinal = potenciarCadena(cadInvertida, absPot);
         free(cadInvertida);
      } else {
         cadFinal = potenciarCadena(cadBase, absPot);
      }

      if (cadFinal != NULL && strlen(cadFinal) > 0) {
         agregaraLista(indiceResultado, cadFinal);
```

```
            }

            free(cadFinal);
            actual = actual->siguiente;
        }
    }

    printf("Resultado de L%d ^ %d generado en L%d.\n", indiceL1, pot, indiceResultado);
    mostrarLista(LResultado->cabeza);
    guardarArchivo(LResultado, "ResultadoPotencia.txt");
}

/* Agrega a una lista de listas nueva hasta la 4ta potencia sin considerar la cadena vacía */
void cerraduraPositiva(int indiceL1) {
    int indiceResultado;

    printf("\n--- Operación: Cerradura Positiva (L+) - Repeticiones w^1 a w^4 ---\n");
    mostrarListasTodas();

    indiceResultado = agregarListaNueva();
    if (indiceResultado == -1) return;
    Lista* LResultado = arrayList[indiceResultado];
    Lista* L1 = arrayList[indiceL1];

    Nodo* actual = L1->cabeza;
    while (actual != NULL) {

        for (int k = 1; k <= 4; k++) {
            char* cadPotencia = potenciarCadena(actual->dato, k);

            if (cadPotencia != NULL && strlen(cadPotencia) > 0) {
                agregaraLista(indiceResultado, cadPotencia);
            }
            free(cadPotencia);
        }
```

```
        actual = actual->siguiente;
    }

    printf("Resultado de L%d+ (repeticiones w^1 a w^4) generado en L%d.\n", indiceL1, indiceResultado);
    mostrarLista(LResultado->cabeza);
    guardarArchivo(LResultado, "ResultadoCerraduraPositiva.txt");
}

/* Esta función hace la unión de todas las potencias mostradas de la cadena y las agrega a una lista nueva en la lista de listas
  L* = L^0 U L+ */
void cerraduraKleane(int indiceL1) {
    int indiceResultado;

    printf("\n--- Operacion: Cerradura de Kleene (L*) - Repeticiones w^0 a w^4 ---\n");
    mostrarListasTodas();

    indiceResultado = agregarListaNueva();
    if (indiceResultado == -1) return;
    Lista* LResultado = arrayList[indiceResultado];

    agregaraLista(indiceResultado, "λ");

    Lista* L1 = arrayList[indiceL1];

    Nodo* actual = L1->cabeza;
    while (actual != NULL) {
        for (int k = 1; k <= 4; k++) {
            char* cadPotencia = potenciarCadena(actual->dato, k);

            if (cadPotencia != NULL && strlen(cadPotencia) > 0) {
                agregaraLista(indiceResultado, cadPotencia);
            }
            free(cadPotencia);
        }
        actual = actual->siguiente;
```

```
    }

    printf("Resultado de L%d* (repeticiones w^0 a w^4) generado en L%d.\n", indiceL1, indiceResultado);
    mostrarLista(LResultado->cabeza);
    guardarArchivo(LResultado, "ResultadoCerraduraKleene.txt");
}

/* Agrega una lista nueva a la lista de listas donde invierte los elementos de la lista elegida por el usuario*/
void reflexion(int indiceL1) {
    int indiceResultado;

    printf("\n--- Operacion: Reflexion (L^R) ---\n");
    mostrarListasTodas();

    indiceResultado = agregarListaNueva();
    if (indiceResultado == -1) return;
    Lista* LResultado = arrayList[indiceResultado];
    Lista* L1 = arrayList[indiceL1];

    // 2. Invertir cada elemento de L1 y agregarlo a LResultado
    Nodo* actual = L1->cabeza;
    while (actual != NULL) {
        // Crear una copia de la cadena para poder invertirla
        char* cadNueva = strdup(actual->dato);
        if (cadNueva == NULL) {
            perror("Error de memoria en reflexion");
            return;
        }

        invertirCadena(cadNueva);
        agregaraLista(indiceResultado, cadNueva); //Agrega la cadena invertida
        free(cadNueva);

        actual = actual->siguiente;
    }
```

```
    printf("Resultado de L%d^R generado en L%d.\n", indiceL1, indiceResultado);
    mostrarLista(LResultado->cabeza);
    guardarArchivo(LResultado, "ResultadoReflexion.txt");
}

void menu(int indiceActual) {
    int resp;
    int ind;

    if (indiceActual < 0 || indiceActual >= numListas) {
        printf("\nError: Indice de lenguaje invalido. Saliendo del menu.\n");
        return;
    }

    printf("\n\n\n\t Trabajando con L%d. Elige una opcion: ", indiceActual);
    printf("\n 1. Union");
    printf("\n 2. Concatenacion");
    printf("\n 3. Potencia");
    printf("\n 4. Cerradura Positiva (L+)");
    printf("\n 5. Cerradura de Kleene (L*)");
    printf("\n 6. Reflexion (L^R)");
    printf("\n 7. Cambiar Lenguaje"); // Opción para cambiar de lenguaje
    printf("\n 8. Salir");

    printf("\nOperacion que desea realizar: ");

    if (scanf("%d", &resp) != 1) {
        printf("Entrada invalida. Intente de nuevo.\n");
        limpiarBuffer();

        menu(indiceActual);
        return;
    }
    limpiarBuffer();
```

```
printf("\n");

switch(resp){

   case 1:
      unioN(indiceActual);
      mostrarListasTodas();
      menu(indiceActual);
   break;

   case 2:
      concatenacion(indiceActual);
      mostrarListasTodas();
      menu(indiceActual);
   break;

   case 3:
      potencia(indiceActual);
      mostrarListasTodas();
      menu(indiceActual);
   break;

   case 4:
      cerraduraPositiva(indiceActual);
      mostrarListasTodas();
      menu(indiceActual);
   break;

   case 5:
      cerraduraKleane(indiceActual);
      mostrarListasTodas();
      menu(indiceActual);
   break;
```

```
        case 6:
            reflexion(indiceActual);
            mostrarListasTodas();
            menu(indiceActual);
        break;

        case 7:
            mostrarListasTodas();
            printf("Con que lenguaje desea trabajar (L): ");
            // Leer el nuevo índice
            if (scanf("L%d", &ind) == 1 && ind >= 0 && ind < numListas) {
                menu(ind);
            } else {
                printf("Indice de lenguaje no valido. Volviendo al menu anterior.\n");
                limpiarBuffer();
                menu(indiceActual);
            }
        break;

        case 8:
            printf("Vuelve pronto! :)\n\n");
        break;

        default:
            printf("Opcion no valida, ingrese una del menu...");
            menu(indiceActual);
        break;

    }
}

void limpiarBuffer() {
    int c;
    while ((c = getchar()) != '\n' && c != EOF);
}
```

```
int main(){

    inicializarArreglo();

    char nombreArch1[50], nombreArch2[50], nombreArch3[50];
    char* archivoCadenas = NULL;
    int indice;
    int len;



    printf("\n Ingrese la ubicacion del primer archivo: ");
    if (scanf("%49s", nombreArch1) != 1) return 0;
    archivoCadenas = informacionArchivo(nombreArch1);
    indice = agregarListaNueva();

    if(archivoCadenas != NULL && indice != -1){
       procesarPalabras(indice, archivoCadenas);
       free(archivoCadenas);
       archivoCadenas = NULL;
    }

    printf("\n Ingrese la ubicacion del segundo archivo: ");
    if (scanf("%49s", nombreArch2) != 1) return 0;
    archivoCadenas = informacionArchivo(nombreArch2);
    indice = agregarListaNueva();

    if(archivoCadenas != NULL && indice != -1){
       procesarPalabras(indice, archivoCadenas);
       free(archivoCadenas);
       archivoCadenas = NULL;
    }

    printf("\n Ingrese la ubicacion del tercer archivo: ");
    if (scanf("%49s", nombreArch3) != 1) return 0;
```

```
    archivoCadenas = informacionArchivo(nombreArch3);
    indice = agregarListaNueva();

    if(archivoCadenas != NULL && indice != -1){
        procesarPalabras(indice, archivoCadenas);
        free(archivoCadenas);
        archivoCadenas = NULL;
    }

    limpiarBuffer();

    mostrarListasTodas();

    limpiarBuffer();

    printf("\n Eliga un lenguaje a utilizar (L): ");

    if (scanf("L%d", &len) == 1 && len >= 0 && len < numListas) {
        menu(len);
    } else {
        printf("Indice de lenguaje no valido. Saliendo.\n");
    }
    liberarMemoria();
    return 0;
}
```

Resultados

```
 Ingrese la ubicacion del primer archivo: C:\pruebas\c1.txt
Lista agregada: L0 inicializada.

 Ingrese la ubicacion del segundo archivo: C:\pruebas\c2.txt
Lista agregada: L1 inicializada.

 Ingrese la ubicacion del tercer archivo: C:\pruebas\c3.txt
Lista agregada: L2 inicializada.

         Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
```

```
 Eliga un lenguaje a utilizar (L): L0



        Trabajando con L0. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 1


--- Operacion: Union ---

        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}

El lenguaje L0 sera el primero.
Ingresa el indice del segundo lenguaje (L): L1
Lista agregada: L3 inicializada.
Resultado de L0 U L1 generado en L3.
Resultado: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
Lista guardada en el archivo: ResultadoUnion.txt

        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
```

```
        Trabajando con L0. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 2


--- Operacion: Concatenaci├│n ---

        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}

El lenguaje L0 sera el primero.
Ingresa el indice del segundo lenguaje (L): L2
Lista agregada: L4 inicializada.
Resultado de L0 . L2 generado en L4.
Resultado: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, am
arillohexagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexag
ono, rosacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexago
no, moradocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohe
xagono}
Lista guardada en el archivo: ResultadoConcatenacion.txt
```

```
        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}




        Trabajando con L0. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 3


--- Operacion: Potencia (L^n) ---

        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
```

```
Lenguaje a potenciar: L0
Ingresa el valor de la potencia (rango: -5 a 10): 3
Lista agregada: L5 inicializada.
Resultado de L0 ^ 3 generado en L5.
Resultado: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
Lista guardada en el archivo: ResultadoPotencia.txt

        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}




        Trabajando con L0. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 3


--- Operacion: Potencia (L^n) ---
```

```
        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}

Lenguaje a potenciar: L0
Ingresa el valor de la potencia (rango: -5 a 10): -3
Lista agregada: L6 inicializada.
Resultado de L0 ^ -3 generado en L6.
Resultado: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
Lista guardada en el archivo: ResultadoPotencia.txt

        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
```

```
        Trabajando con L0. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 4


--- Operaci├│n: Cerradura Positiva (L+) - Repeticiones w^1 a w^4 ---

        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}

Lista agregada: L7 inicializada.
Resultado de L0+ (repeticiones w^1 a w^4) generado en L7.
Resultado: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojoro
jo, rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjana
ranja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblan
coblanco, blancoblancoblancoblanco}
Lista guardada en el archivo: ResultadoCerraduraPositiva.txt
```

```
        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
L7: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, roj
orojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja,
naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanc
o, blancoblancoblancoblanco}




        Trabajando con L0. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 5


--- Operacion: Cerradura de Kleene (L*) - Repeticiones w^0 a w^4 ---
```

```
        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
L7: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, roj
orojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja,
naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanc
o, blancoblancoblancoblanco}

Lista agregada: L8 inicializada.
Resultado de L0* (repeticiones w^0 a w^4) generado en L8.
Resultado: {╬, azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, ro
jorojo, rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naran
janaranja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blanco
blancoblanco, blancoblancoblancoblanco}
Lista guardada en el archivo: ResultadoCerraduraKleene.txt
```

```
        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
L7: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, roj
orojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja,
naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanc
o, blancoblancoblancoblanco}
L8: {╬, azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo,
 rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaran
ja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancob
lanco, blancoblancoblancoblanco}




        Trabajando con L0. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 6


--- Operacion: Reflexion (L^R) ---
```

```
        Listas disponibles:
 L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
 L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
 L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
 L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
 L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
 exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
 sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
 adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
 L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
 L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
 L7: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, roj
 orojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja,
 naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanc
 o, blancoblancoblancoblanco}
 L8: {╬, azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo,
  rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaran
 ja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancob
 lanco, blancoblancoblancoblanco}

 Lista agregada: L9 inicializada.
 Resultado de L0^R generado en L9.
 Resultado: {luza, ollirama, ojor, edrev, asor, ajnaran, odarom, ocnalb}
 Lista guardada en el archivo: ResultadoReflexion.txt

        Listas disponibles:
 L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
 L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
 L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
 L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
 L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
 exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
 sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
 adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
 L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
 L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
 L7: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, roj
 orojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja,
 naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanc
 o, blancoblancoblancoblanco}
```

```
L8: {╬, azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo,
 rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaran
ja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancob
lanco, blancoblancoblancoblanco}
L9: {luza, ollirama, ojor, edrev, asor, ajnaran, odarom, ocnalb}




        Trabajando con L0. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 7
```

```
        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
L7: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, roj
orojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja,
naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanco
o, blancoblancoblancoblanco}
L8: {╬, azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo,
 rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaran
ja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancob
lanco, blancoblancoblancoblanco}
L9: {luza, ollirama, ojor, edrev, asor, ajnaran, odarom, ocnalb}

Con que lenguaje desea trabajar (L): L1



        Trabajando con L1. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 1
```

```
--- Operacion: Union ---

        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarilloh
exagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, ro
sacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, mor
adocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
L7: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, roj
orojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja,
naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanco
o, blancoblancoblancoblanco}
L8: {╬, azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo,
 rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaran
ja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancob
lanco, blancoblancoblancoblanco}
L9: {luza, ollirama, ojor, edrev, asor, ajnaran, odarom, ocnalb}

El lenguaje L1 sera el primero.
Ingresa el indice del segundo lenguaje (L): L3
Lista agregada: L10 inicializada.
Resultado de L1 U L3 generado en L10.
Resultado: {perro, gato, conejo, tortuga, mariposa, raton, leon, azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
Lista guardada en el archivo: ResultadoUnion.txt
```

```
        Listas disponibles:
L0: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}
L1: {perro, gato, conejo, tortuga, mariposa, raton, leon}
L2: {cuadrado, circulo, triangulo, rectangulo, hexagono}
L3: {azul, amarillo, rojo, verde, rosa, naranja, morado, blanco, perro, gato, conejo, tortuga, mariposa, raton, leon}
L4: {azulcuadrado, azulcirculo, azultriangulo, azulrectangulo, azulhexagono, amarillocuadrado, amarillocirculo, amarillotriangulo, amarillorectangulo, amarillohexagono, rojocuadrado, rojocirculo, rojotriangulo, rojorectangulo, rojohexagono, verdecuadrado, verdecirculo, verdetriangulo, verderectangulo, verdehexagono, rosacuadrado, rosacirculo, rosatriangulo, rosarectangulo, rosahexagono, naranjacuadrado, naranjacirculo, naranjatriangulo, naranjarectangulo, naranjahexagono, moradocuadrado, moradocirculo, moradotriangulo, moradorectangulo, moradohexagono, blancocuadrado, blancocirculo, blancotriangulo, blancorectangulo, blancohexagono}
L5: {azulazulazul, amarilloamarilloamarillo, rojorojorojo, verdeverdeverde, rosarosarosa, naranjanaranjanaranja, moradomoradomorado, blancoblancoblanco}
L6: {luzaluzaluza, olliramaolliramaollirama, ojorojorojor, edrevedrevedrev, asorasorasor, ajnaranajnaranajnaran, odaromodaromodarom, ocnalbocnalbocnalb}
L7: {azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanco, blancoblancoblancoblanco}
L8: {╬, azul, azulazul, azulazulazul, azulazulazulazul, amarillo, amarilloamarillo, amarilloamarilloamarillo, amarilloamarilloamarilloamarillo, rojo, rojorojo, rojorojorojo, rojorojorojorojo, verde, verdeverde, verdeverdeverde, verdeverdeverdeverde, rosa, rosarosa, rosarosarosa, rosarosarosarosa, naranja, naranjanaranja, naranjanaranjanaranja, naranjanaranjanaranjanaranja, morado, moradomorado, moradomoradomorado, moradomoradomoradomorado, blanco, blancoblanco, blancoblancoblanco, blancoblancoblancoblanco}
L9: {luza, ollirama, ojor, edrev, asor, ajnaran, odarom, ocnalb}
L10: {perro, gato, conejo, tortuga, mariposa, raton, leon, azul, amarillo, rojo, verde, rosa, naranja, morado, blanco}



        Trabajando con L1. Elige una opcion:
 1. Union
 2. Concatenacion
 3. Potencia
 4. Cerradura Positiva (L+)
 5. Cerradura de Kleene (L*)
 6. Reflexion (L^R)
 7. Cambiar Lenguaje
 8. Salir
Operacion que desea realizar: 8

Vuelve pronto! :)
```

ResultadoCerraduraKleene	16/10/2025 19:54	Documento de tex...	1 KB
ResultadoCerraduraPositiva	16/10/2025 19:54	Documento de tex...	1 KB
ResultadoConcatenacion	16/10/2025 19:53	Documento de tex...	1 KB
ResultadoPotencia	16/10/2025 19:54	Documento de tex...	1 KB
ResultadoReflexion	16/10/2025 19:54	Documento de tex...	1 KB
ResultadoUnion	16/10/2025 19:55	Documento de tex...	1 KB