Estrutura de Dados: (Programação funcional)

- A estrutura de dados utilizada é uma lista em SML, que é uma linguagem funcional. Listas são coleções sequenciais de elementos onde cada elemento possui um sucessor (exceto o último) e podem conter elementos de diferentes tipos

Algoritmos para Operação:

- Verificar se a lista é vazia (ehVazio):
  - Utilização da função NULL para verificar se é vazia

- União de duas listas (uniaoConj):
  - Concatena a lista A com os elementos de B que não estão em A

- Interseção de duas listas (interseConj):
  - Filtra os elementos de A que também estão em B

- Diferença entre duas listas (diffConj):
  - Filtra os elementos de A que não estão em B

- Diferença Simétrica entre duas listas (diffSimConj):
  - Concatena a diferença de A em relação a B com a diferença de B em relação a A

- Adicionar elemento a uma lista (addConj):

  - Adiciona o elemento a lista se não houver

- Remover Elemento de uma lista (remConj):

  - filtra os elementos diferentes do elemento a ser removido

- Tamanho de uma lista (tamConj):

  - Usa a função length para obter o tamanho da lista

- Verificar se um elemento pertence a uma lista (ehElemConj):

  - Usa a função list.exist para verificar a presença do elemento na lista

- Utilização:

- Entrada de Dados: A e B devem ser definidos

- Operações: Chame as funções desejadas, passando argumentos

```
(* Exemplo de uso *)
(*val conjunto_A = [1, 2, 3, 4, 5];
val conjunto_B = [4, 5, 6, 7, 8];

(* Imprimindo os resultados *)
print ("É vazio (A): " ^ Bool.toString (ehVazio(conjunto_A)) ^ "\n");
print ("União de A e B: " ^ Int.toString (uniaoConj(conjunto_A, conjunto_B)) ^ "\n");
print ("Interseção de A e B: " ^ Int.toString (intersConj(conjunto_A, conjunto_B)) ^ "\n");
print ("Diferença A - B: " ^ Int.toString (diffConj(conjunto_A, conjunto_B)) ^ "\n");
print ("Diferença Simétrica A ⊕ B: " ^ Int.toString (diffSimConj(conjunto_A, conjunto_B)) ^ "\n");

(* Adicionando um elemento a A *)
val novo_conjunto_A = addConj(conjunto_A, 10);
print ("A após adição: " ^ Int.toString novo_conjunto_A ^ "\n");

(* Removendo um elemento de A *)
val novo_conjunto_A = remConj(conjunto_A, 3);
print ("A após remoção: " ^ Int.toString novo_conjunto_A ^ "\n");

(* Tamanho de A *)
print ("Tamanho de A: " ^ Int.toString (tamConj(conjunto_A)) ^ "\n");

(* Verificando se um elemento pertence a A *)
val elemento_verificar = 2;
print (Int.toString elemento_verificar ^ " pertence a A: " ^ Bool.toString
(ehElemConj(conjunto_A, elemento_verificar)) ^ "\n");*)
```