https://github.com/alexteo24/University-Work/tree/main/ThridYear/LFTC/Lab8/Compiler/src

As a data structure for the symbol table, I chose to implement a generic BST.

It's inner class, Node, will contain as attributes data of type T and left and right of type Node.

The BST has as attributes only the head of type Node, which will initially be null.

```
//The function will add data to the BST if the data is not already present
//data: the information we want to add to our BST
//return -
public void add(T data)
```

```
//The function will search for some data.
//data: the data we want to seach in our BST
//return: if the data is present, return the data or null otherwise
public T find(T data)
```

```
//Constructs and returns string form of the BST
public String displayTree()
```

As a data structure for the pif I had to implement a Linked list.
It's inner class, Node, will contain as attributes data of type T and next of type Node.

The linked list has as attributes only the head of type Node, which will initially be null.

```
//The function will add data to the linked list
//data: the information we want to add to our linked list
//return -
public void add(T data)
```

```
//Constructs and returns string form of the linked list
public String displayList()
```

Functions for the LR(0) parser

```
// Function for the LR(0) parser which will enrich the grammer by introducing a new non-terminal
// which will have a production to the current starting symbol. The newly introduced production will
// become the starting symbol.
// Once the new non-terminal was introduced, the canonical set will be built starting from the initial
// and computing the goTo for it and every new resulting state.
public void enrichGrammar()
```

```
// Computes the closure for a given list of productions of form %s -> %s
public List<String> closure(List<String> productions)
```

```
// Computes the go for a state and a given token
public List<String> goTo(List<String> state, String token)
```

EXAMPLE

Enriched grammar
SR
A S SR
a b c
A -> bA | c
S -> aA
SR -> .S

Canonical set
[[SR->S.], [S->aA.], [SR->.S, S->.aA], [A->c.], [S->a.A, A->.bA, A->.c], [A->b.A, A->.bA, A->.c], [A->bA.]]