Proj-part2 – Course number 0460266

Code Explanation

Dor Yefet – 208017657

Nadav Zohar – 206457343

<u>Lex file (part2.lex):</u>

We used Part1 of the Project as the lex file, with few modifications: Each rule now creates a new tree node by calling the makeNode function with the relevant values and assigns it to the yylval argument. The return value of each rule is the corresponding named token. For example:

```
{elipsis}                    { yylval = makeNode(yytext, NULL, NULL); return tk_elipsis; }
{relop}                      { yylval = makeNode("relop", yytext, NULL); return tk_relop; }
```

The lex parser output goes to the bison interpreter, that declares the token in their precedence order.

<u>Bison file (part2.y):</u>

We declared a global variable for the parser tree root (parseTree), which will later be assigned to the first symbol in the hierarchy (the PROGRAM symbol).
After defining all the tokens, we implemented the grammar of the language as specified in the provided work PDF. Each rule is constructed using the defined tokens and symbols. Within the actions for these rules, a child node is created for each symbol, and subsequent tokens, literals, or symbols in the rule are added as sibling nodes using the concatList function.
This approach ensures that the parse tree is generated in the correct hierarchical order, with the root node, child nodes, and sibling relationships properly defined.