# VERTEX PROGRAMMING LANGUAGE

Team 2
1] Amulya Bodla (abodla)
2] Niharika Pothana (npothana)
3] Piyush Mudireddy (prmudire)
4] Ritesh Reddy(ranugu)

# Glossary

- Overview & Tools Used
- Features
- Workflow
- Lexer
- Grammar
- Compiler
- Intermediate Code Translation
- Sample Code
- Runtime
- Additional Functionalities

# Overview & Tools Used

- **Lexical Analysis and parsing: Prolog**
- **Intermediate code generation:**

  **FileName: Intermediate code**

  **Extension: .intc**
- **Runtime environment: Prolog**
- **Language extension: .vtex**

# Features

**Arithmetic Operators**

**Primitive Data Types i.e. integer, Boolean, String.**

**Boolean operations (AND, OR, NOT)**

**Assignment Operator**

**Relation operations (>, <, >=, <=, ==, ~)**

**Conditional Statements (if-then, if-then-else, ternary)**

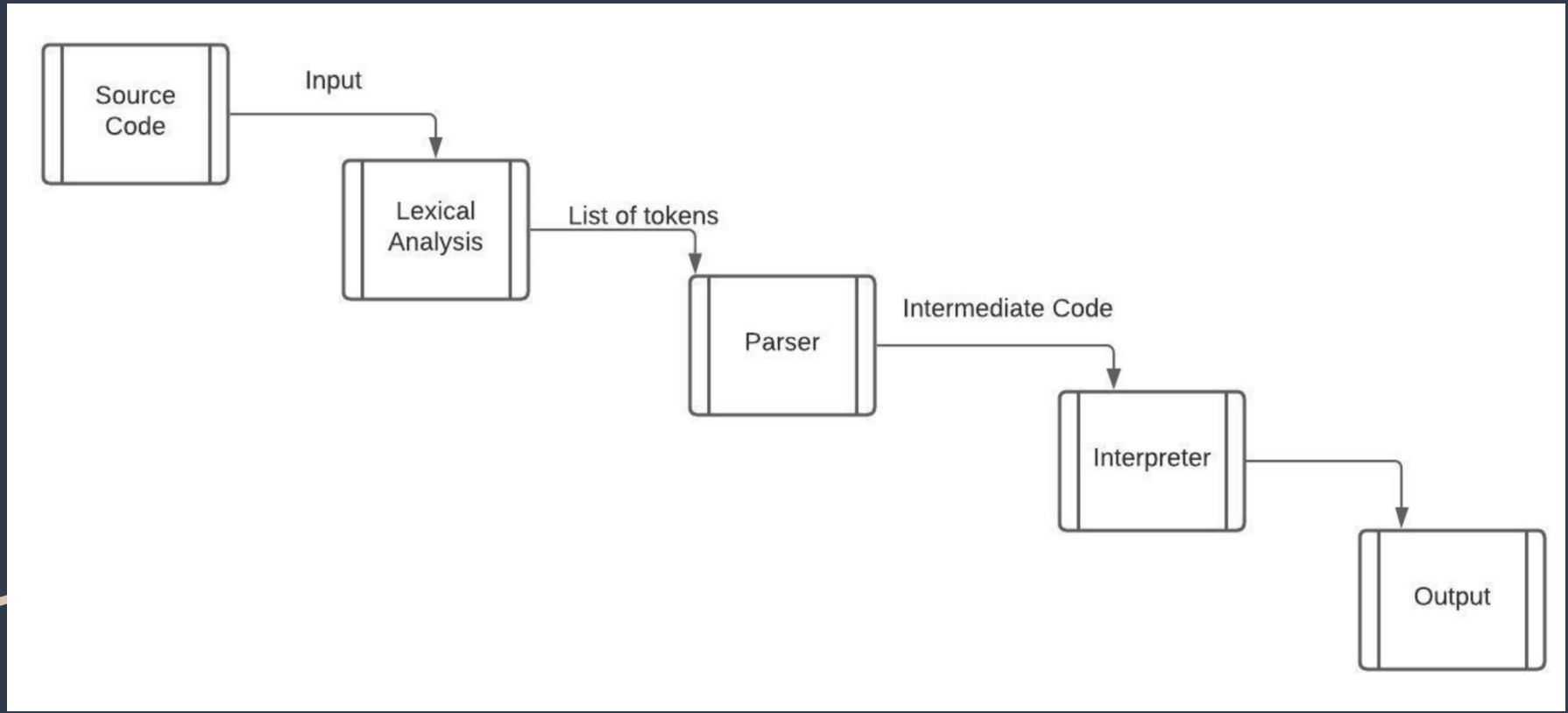
**Display & DisplayIn**

**Iterative Statements (for, while, for in range)**

**Low Level Intermediate Language (.intc files)**

**Block Structured**

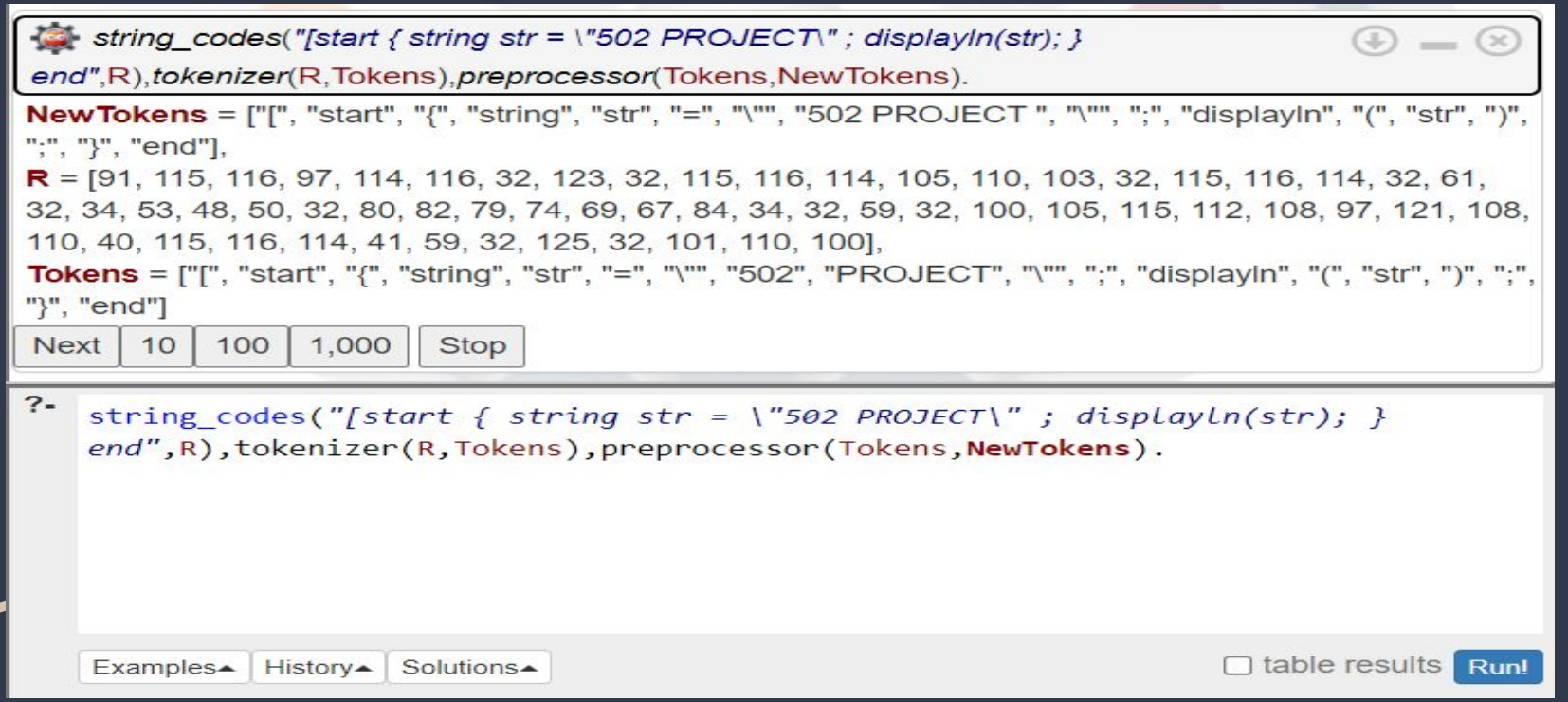
# Workflow


Source Code
Input
Lexical Analysis
List of tokens
Parser
Intermediate Code
Interpreter
Output

# Lexer

```
tokenizer([],[]).
tokenizer([Code|Rem],Tokens):-char_type(Code,space),tokenizer(Rem,Tokens),!.
tokenizer([Code|Codes],[Strings|Tokens]):-char_type(Code,alnum), wordSplit([Code|Codes],Words,Rem),
    name(Word,Words), atom_string(Word,Strings), tokenizer(Rem,Tokens),!.
tokenizer([Code|Rem],[Strings|Tokens]):-name(Char,[Code]), atom_string(Char,Strings), tokenizer(Rem,Tokens).

wordSplit([Code1,Code2|Rem],[Code1|Words],Res):-char_type(Code2,alnum), wordSplit([Code2|Rem],Words,Res).
wordSplit([Code1|Rem],[Code1],Rem).

preprocessor([],[]).
preprocessor([H|T],[H,Str,H1|R1]) :- H = "\"" , processor(T,[H1|T1],R),atom_string(R,Str),
    preprocessor(T1,R1).
preprocessor([H|T],[H|R]) :- H \= "\"" , preprocessor(T,R).

processor([H|T],L,R) :- H\= "\"",atom_string(H,H1),processor(T,L,R1),string_concat(H1," ",R2),
    string_concat(R2,R1,R).
processor([H|T],[H|T],"") :- H = "\"".
```

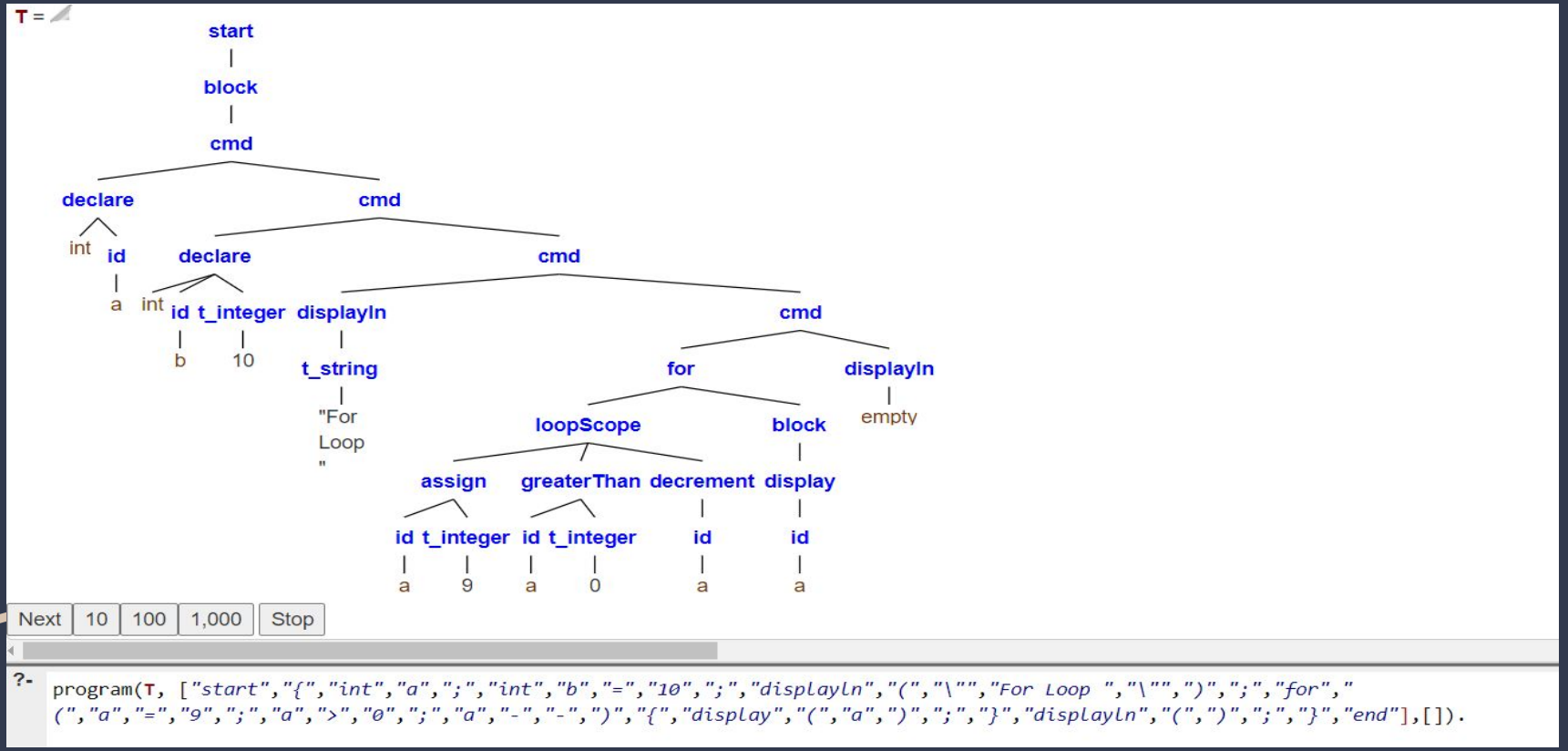
# Lexer


string_codes("[start { string str = \"502 PROJECT\" ; displayln(str); } end",R),tokenizer(R,Tokens),preprocessor(Tokens,NewTokens).
NewTokens = ["[", "start", "{", "string", "str", "=", "\"", "502 PROJECT ", "\"", ";", "displayln", "(", "str", ")", ";", "}", "end"],
R = [91, 115, 116, 97, 114, 116, 32, 123, 32, 115, 116, 114, 105, 110, 103, 32, 115, 116, 114, 32, 61, 32, 34, 53, 48, 50, 32, 80, 82, 79, 74, 69, 67, 84, 34, 32, 59, 32, 100, 105, 115, 112, 108, 97, 121, 108, 110, 40, 115, 116, 114, 41, 59, 32, 125, 32, 101, 110, 100],
Tokens = ["[", "start", "{", "string", "str", "=", "\"", "502", "PROJECT", "\"", ";", "displayln", "(", "str", ")", ";", "}", "end"]
Next
10
100
1,000
Stop
?-
string_codes("[start { string str = \"502 PROJECT\" ; displayln(str); } end",R),tokenizer(R,Tokens),preprocessor(Tokens,NewTokens).
Examples
History
Solutions
table results
Run!

# Grammar

```
program → START block END.
block → { commands_list }.
commands_list → commands ; commands_list | commands ; .

commands → display | displayln | declare | assign | if-then | if-then-else | while | for | for-in-range | ternary | expr | block .
declare → DATATYPE identifier |  DATATYPE identifier = value.
assign → identifier ASSIGNOP expr | identifier ASSIGNOP ternary | identifier++ | identifier--.
if-else → IF ( compositeBool ) THEN block | IF ( compositeBool ) THEN block ELSE block.

value → string | number | true | false.

for → FOR (loopScope) block| FOR VARCHAR IN RANGE ( NUMBERS,NUMBERS ) block.
loopScope →  assign DELIMITER compositeBool DELIMITER assign.
while → WHILE ( compositebool ) block.

ternary → ( compositebool ) TERNARYOP expr TERNARYOP1 expr.

display → DISPLAY expr|DISPLAY member.
compositeBool → bool_expr | bool_expr AND compositeBool | bool_expr OR compositeBool | expr.
bool_expr → expr | expr COMPAREOP expr | NOT expr | BOOL.
expr → component ADDSUB expr | component.
component → member MULDIV component | member .
member → NUMBERS|VARCHAR.
```

# Parse Tree


T =
start
block
cmd
declare
cmd
int
id
declare
cmd
a
int
id t_integer
displayln
cmd
b
10
t_string
for
displayln
"For Loop "
loopScope
block
empty
assign
greaterThan
decrement
display
id t_integer
id t_integer
id
id
a
9
a
0
a
a
Next
10
100
1,000
Stop
?-
program(T, ["start","{","int","a",";","int","b","=","10",";","displayln","(","\"","For Loop ","\"",")",";","for","
("","a","=","9",";","a",">","0",";","a","-","-",")","{","display","(","a",")",";","}","displayln","(",")",";","}","end"],[]).

# Additional Features implemented

**Print variations (display, displayln , expression evaluation)**

**Increment and Decrement operators (++, --)**

**String operations: Concatenation & Reverse functions**

# Interpreter

```
% Evaluation expressions --------------------------------------------

program_eval(start(T)) :-
    eval_block(T,_,_).
eval_block(block(T),Env,NewEnv) :- eval_command_list(T,Env,NewEnv),!.

% Evaluations of various commands.
eval_command_list(cmd(T1,T2),Env,NewEnv) :- eval_command(T1,Env,Env1),eval_command_list(T2,Env1,NewEnv).
eval_command_list(T,Env,NewEnv) :- eval_command(T,Env,NewEnv).
```

```
eval_command(declare(T1,T2),Env,NewEnv) :- eval_id(T2,Id),defaultValue(T1,Val),insert(Id,T1,Val,Env,NewEnv).
eval_command(declare(T1,T2,T3),Env,NewEnv) :- eval_id(T2,Id), eval_value(T3,Val),insert(Id,T1,Val,Env,NewEnv).
eval_command(assign(T1,T2),Env,NewEnv):- eval_expr(T2,Env,Env1,Res1),eval_id(T1,Id), update(Id,Res1,Env1,NewEnv).
eval_command(assign(T1,T2),Env,NewEnv):- eval_ternary(T2,Env,Env1,Res1),eval_id(T1,Id), update(Id,Res1,Env1,NewEnv).
eval_command(increment(T),Env,NewEnv):- eval_increment(T,Env,NewEnv,_).
eval_command(decrement(T),Env,NewEnv):- eval_decrement(T,Env,NewEnv,_).
```

# Interpreter

```
48    eval_command(display(T),Env,Env):- eval_expr(T,Env,Env,Res),write(Res).
49    eval_command(displayln(empty),Env,Env):-writeln("").
50    eval_command(displayln(T),Env,Env):- eval_expr(T,Env,Env,Res),writeln(Res).
```

```
% Expressions evaluation.
eval_expr(expr_assign(T1,T2),Env,NewEnv,Res):- eval_expr(T2,Env,Env1,Res),eval_id(T1,Id), update(Id,Res,Env1,NewEnv).
eval_expr(t_string_reverse(T),Env,Env,Res):- eval_expr(T,Env,Env,Str),string(Str),string_to_list(Str,L),reverse(L,Rev),string_to_list(Res,Rev).
eval_expr(t_string_concat(T1,T2),Env,Env,Res) :- eval_expr(T1,Env,Env,R1),eval_expr(T2,Env,Env,R2),string(R1),string(R2),string_concat(R1,R2,Res).
eval_expr(increment(T),Env,NewEnv,Res):- eval_increment(T,Env,NewEnv,Res).
eval_expr(decrement(T),Env,NewEnv,Res):- eval_decrement(T,Env,NewEnv,Res).
```

# Interpreter

```
% Lookup predicate to check the environment for variable values
lookup(Id,[],_):- write(Id),write(' not found'),fail.
lookup(Id,[(Id,_,Val)|_],Val).
lookup(Id1,[(Id2,_,_)|Env],Res):- Id1 \= Id2, lookup(Id1,Env,Res).

% update predicate to update a value in the environment
update(Id,_,[],_):- write(Id),write(' not declared'),fail.
update(Id,Val,[(Id,Type,_)|T],[(Id,Type,Val)|T]):- checkTypeValue(Type,Val,valid).
update(Id,Val,[(Id,Type,OldVal)|T],[(Id,Type,OldVal)|T]):- checkTypeValue(Type,Val,invalid),write('You are trying to update invalid value for '),writeln(Id),!,fail.
update(Id,Val,[H|T],[H|R]) :- H \= (Id,_), update(Id, Val, T, R).

checkTypeValue(int,Val,valid) :- integer(Val).
checkTypeValue(int,Val,invalid) :- \+ integer(Val).
checkTypeValue(bool,true,valid).
checkTypeValue(bool,false,valid).
checkTypeValue(bool,E,invalid) :- E\= true ; E\= false.
checkTypeValue(string,Val,valid) :- string(Val).
checkTypeValue(string,Val,invalid) :- \+ string(Val).

% Insert
insert(Id,Type,Val,[],[(Id,Type,Val)]) :- checkTypeValue(Type, Val , valid),!.
insert(_Id,Type,Val,E,E) :- checkTypeValue(Type, Val , invalid), writeln('Invalid value assignment'),!,fail.
insert(Id,Type,Val,[H|T],[H|R]) :- insert(Id, Type, Val, T, R).
```

# Sample Input

```
start
{
int a;
int b=10;
displayln("For Loop");
for(a=9;a>0;a--){
    display(a);
}
displayln();
display("For Range output -> ");
for(a in range (11,20)){
    display(a);
    display(", ");
}
displayln();
display("While loop output -> ");
while(a<30){
    display(a);
    display(", ");
    a++;
}
displayln();
displayln("If Then");
if(a<b)then{
    display("Ifthen success");
}
display("If Else's if output -> ");

string str = "gnirts desrever si sihT" ;
displayln(reverse(str));
string str1= "Happy";
string str2= " Times";
display("After concating str1 and str2 we get");
displayln(concat(str1,str2));
}
end
```

# Sample Intermediate Code

```
start(block(cmd(declare(int,id(a)),cmd(declare(int,id(b),t_integer(10)),
cmd(displayln(t_string("For Loop ")),cmd(for(loopScope(assign(id(a),
t_integer(9)),greaterThan(id(a),t_integer(0)),decrement(id(a))),block(display(id(a)))),
cmd(displayln(empty),cmd(display(t_string("For Range output - > ")),
cmd(forRange(id(a),t_integer(11),t_integer(20),block(cmd(display(id(a)),
display(t_string(", "))))),cmd(displayln(empty),cmd(display(t_string("While loop output - > ")),
cmd(while(lessThan(id(a),t_integer(30)),block(cmd(display(id(a)),cmd(display(t_string(", ")),
increment(id(a)))))),cmd(displayln(empty),cmd(displayln(t_string("If Then ")),
cmd(ifThen(lessThan(id(a),id(b)),block(display(t_string("Ifthen success "))))),
cmd(display(t_string("If Else ' s if output - > ")),cmd(ifElse(lesserThanEq(id(a),id(b)),
block(displayln(t_add(id(a),id(b)))),block(displayln(t_sub(id(a),id(b))))),
cmd(display(t_string("If Else ' s else output - > ")),
cmd(ifElse(greaterThan(id(a),id(b)),block(displayln(t_add(id(a),id(b)))),
block(displayln(t_sub(id(a),id(b))))),cmd(declare(string,id(str),
t_string("gnirts desrever si sihT ")),cmd(displayln(t_string_reverse(id(str)))),
cmd(declare(string,id(str1),t_string("Happy ")),cmd(declare(string,id(str2),t_string("Times ")),
cmd(display(t_string("After concating str1 and str2 we get ")),
displayln(t_string_concat(id(str1),id(str2))))))))))))))))))))))))).
```

# Sample Output

```
?- consult('ParseTreeGenerator.pl').
true.

?- consult('Evaluator.pl').
true.

?- vertex('megaSample.vtex').
For Loop
987654321
For Range output -> 11, 12, 13, 14, 15, 16, 17, 18, 19,
While loop output -> 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
If Then
If Else 's if output -> 20
If Else 's else output -> 40
 This is reversed string
After concating str1 and str2 we get Happy Times
true
```

GITHUB LINK: https://github.com/Prmudire/SER502-Spring2021-Team2

# Thank You

Amulya Bodla (1219477220)
Niharika Pothana (1219596937)
Ritesh Reddy (1220005293)
Piyush Reddy (1219456537)