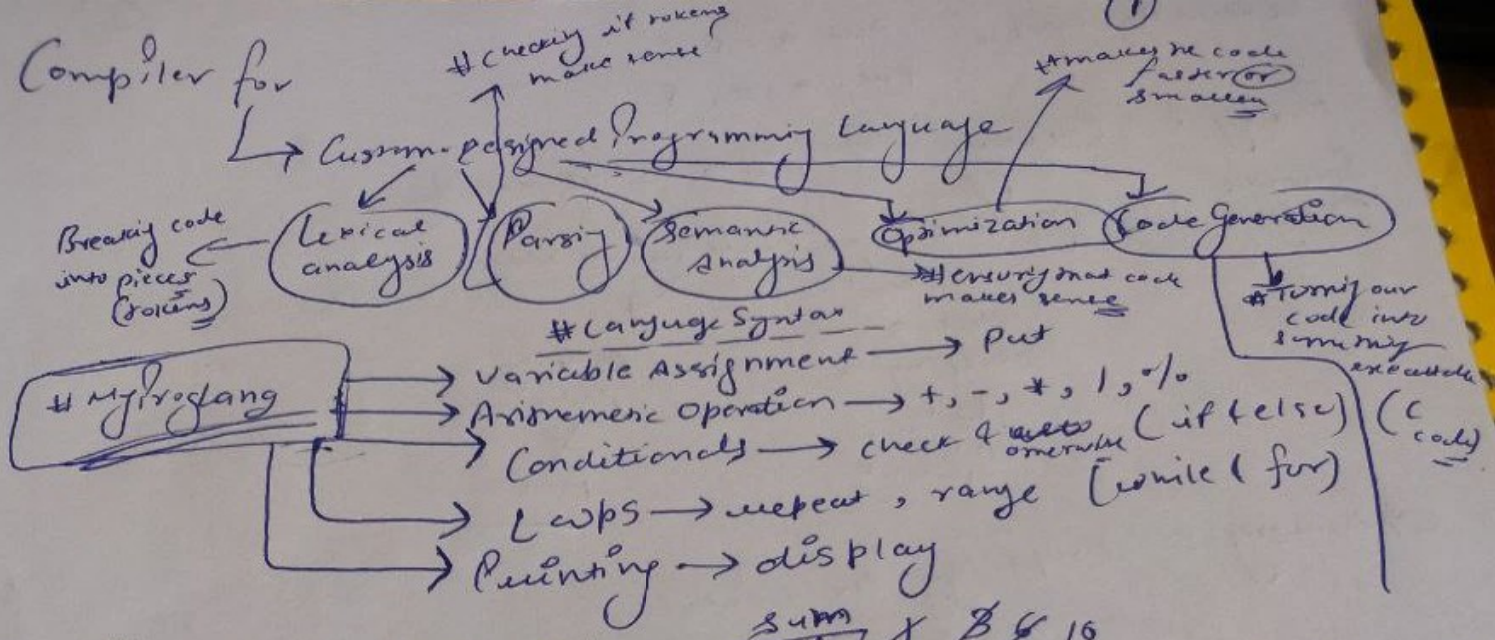
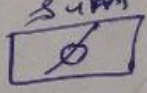



```
put sum = 0;
range i = 1 to 5 {
    put sum = sum + i;
    display sum;
}
```

#Modified (; is not required to end the line)

```
put num = 10
repeat num > 0 {
    put rem = num % 3
    display rem
    put num = num - 1
}
```

sum



1 2 3 4 5 6 10

i = 1 → i = 4 (#for loop)

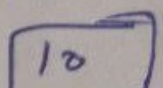
i = 1
sum = sum + 1
= 0 + 1 = 1

i = 2
sum = 1 + 2 = 3

i = 3
sum = 3 + 3 = 6

i = 4
sum = 6 + 4 = 10

num



10 > 0 ✓
rem = 10 % 3 =
10 - 1 = 9
9 > 0
rem = 9 % 3
8 >
- 1
|
|

Lexical Analysis: put a=5
put, a, =, 5

Parsy: Checking if put a=5 is valid

Semantic Analysis: Making sure that code makes sense

put a=5
display a
↳ # makes sense #

display a
put a=10
✗ No sense

Optimizing: Simplifying the code

a = 5 + 0
↳ Simplified ⇒ a = 5

Code Generator: Custom code → C-code → which machine can compile

Project Structure -

lexer.c → for lexical analysis
parser.c → for parsing & generating code
main.c
compiling everything to g
input.txt → Text file for writing language code
output.c → # Generated C co

Parsy
semantic analysis
Optimization
Code Generation

Lexer.c

- Breaks code into tokens
- Tokens are smallest units
- ignore space
- Aspects at delimiters like { , } , (,) ; etc
- Our lexer
  - classifies tokens (keywords, identifiers, numbers, operators, delimeters, strings)
  - Supports string literals & basic error checking
  - # Scans character by character → build tok
    ↓
    assign th
    tokens t

enum
struct

enum : → user defined datatype in C
: allows to assign names to set of integral constants.

enum Color {
Red] automatic // 0
Green] assign // 1
Blue] // 2
};

can a
numb
↓
to call 3
to any ne

enum Color myColor = Red;
%d myColor print → 0

typedef enum {
Mon-
} Days;
Days today = fri

Keywords (special words)
Identifier (var-names)
Number (integers)
Operators
Delimeters
Strings (quoted text)
Unknowns (anything else)

←

Keywords : put display, check otherwise, repeat range

Identifier : Letter (or) underscore — Letters, No.s (or) _

Operators : +, -, *, /, %, =, >, <

Delimeters : ( ) → for cond^ns

{ } → Block delimeters

; → Statement Separator

MID - TERM EVALUATION

Lexer → implicit integer & string support

→ No explicit datatypes su[illegible] like Char or decimals [illegible]

Language : bruno

Component of Lang : Keywords, identifiers, Numbers, Operators, Strings, Delimiters

Data types :

Current ↓

integer (implicit for vars & no.s)

Strings (for display)

Future ↓

Characters

Decimal

User Input.