1. Grammar (Context-Free Grammar)

- start -> dtype MUKH ( ) code
- code -> { lines }
- lines -> statements lines | empty
- statements -> make_var | assign_val | condition | loops | output | jumps | code | ; | error ;
- make_var -> dtype var_list ;
- dtype -> GINTI | ISHARIA | LAFZ | JHANDA
- var_list -> single_var | single_var , var_list
- single_var -> id | id = expression | id [num] | id [num] = { args } | JHANDA | JHANDA = expression
- assign_val -> term_var = expression ; | term_var PlusEqualto expression ; | term_var = { args }
- condition -> JAY (expression) statements | JAY (expression) statements NAHITO statements
- loops -> CHAKKAR (loop_part ; check_part ; loop_part) statements | JADONTAK (expression) statements
- output -> WIKHAO (str) ; | WIKHAO (str , args) ;
- expression -> simple | simple compare simple
- simple -> term | simple + term | simple - term
- term -> factor | term * factor | term / factor | term % factor
- factor -> term_var | num | dec | charecter | str | SACH | JHOOT | (expression)

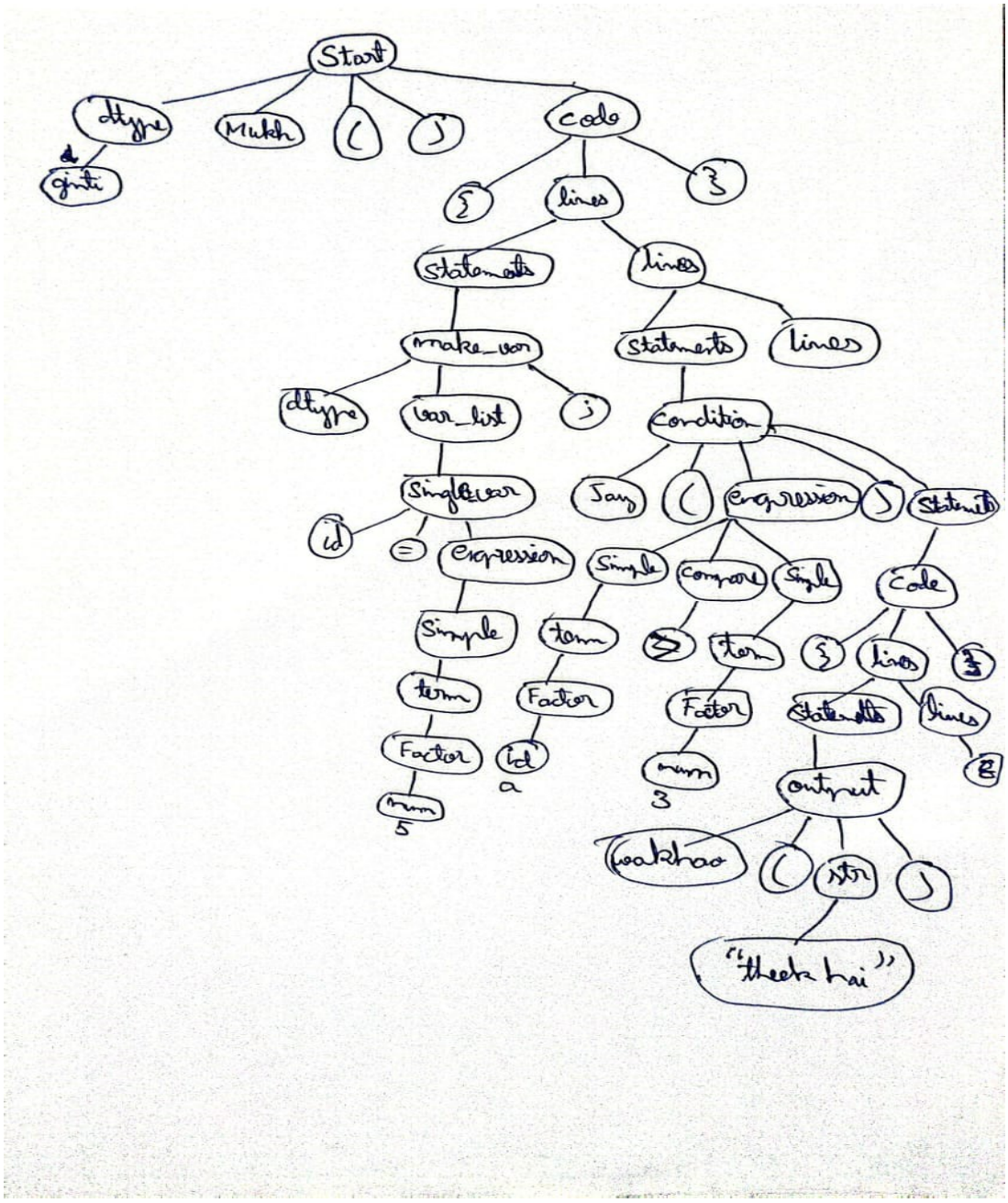
2. FIRST and FOLLOW Sets

Non-Terminal	**FIRST Set**	**FOLLOW Set**
dtype	{ GINTI, ISHARIA, LAFZ, JHANDA }	{ MUKH, id, JHANDA }
statements	{ GINTI, ISHARIA, LAFZ, JHANDA, id, JAY, CHAKKAR, JADONTAK, WIKHAO, MORO, AGAYCHALO, ROKO, {, ; }	{ GINTI, ISHARIA, LAFZ, JHANDA, id, JAY, CHAKKAR, JADONTAK, WIKHAO, MORO, AGAYCHALO, ROKO, {, ;, }, NAHITO }

3. Parse Tree

program code :

GINTI MUKH() { GINTI a = 5; JAY (a > 3) { WIKHAO("Theek hai"); } }



4. Use of Phase 01 Tokens Phase 01 (Lexical Analysis) creates the building blocks for the Parser.

- **Triggering Rules:** Tokens like GINTI or JAY tell the parser exactly which rule to start (like a declaration or a condition).
- **Storage:** Identifiers (id) and values (num) are taken from the tokens and saved into the **Symbol Table** so the program knows variable names.
- **Boundaries:** Symbols like { } and ; tell the parser where a block starts and where a specific instruction ends.
- **Error Detection:** If a token appears where it shouldn't (like a number at the start of a function), the parser uses the token info to show exactly where the error is.