UNIVERSITY OF CENTRAL PUNJAB

Faculty of Information Technology



**Project Phase 2**

**Submitted To:** Sir Asif Farooq

**Name:** Muhammad Hassan Javed

**ROLL N0:** L1F22BSCS0569

**Section:** G-5

**Language Name:** Asaan Bhasha

**Overview:** Asaan Bhasha is a user-friendly, high-level programming language designed to make coding natural and approachable for beginners. Its primary goal is to lower the cognitive barrier to entry by replacing complex technical syntax with everyday Urdu and Hindi vocabulary, allowing learners to focus on logic rather than syntax.

**Keywords:**

| **Keyword** | **Meaning** | **Example** |
| --- | --- | --- |
| **agar** | if | agar (scor>0) |
| **warna** | else | warna boly "N" |
| **bolay** | print/output | bolay "hello" |
| **warna agar** | else if | warna agar (score>0) |
| **Hukam karay** | input | batao x |
| **thehr jao** | stop/break | Thehrjao |
| **ank** | int | Ank 15 |
| **shabt** | string | "Hassan" |
| **jabtuk** | while | jabtuk(x<5) |
| **Han naa** | bool | Han naa = yes |
| **Shudh** | true | agar Shudh) |
| **jhoot** | false | agar (jhoot) |
| **lotaao** | return | lotaao x |
| **challay** | for loop | challay (x=0; x<2; x++) |
| **karo** | do | karo (x) |

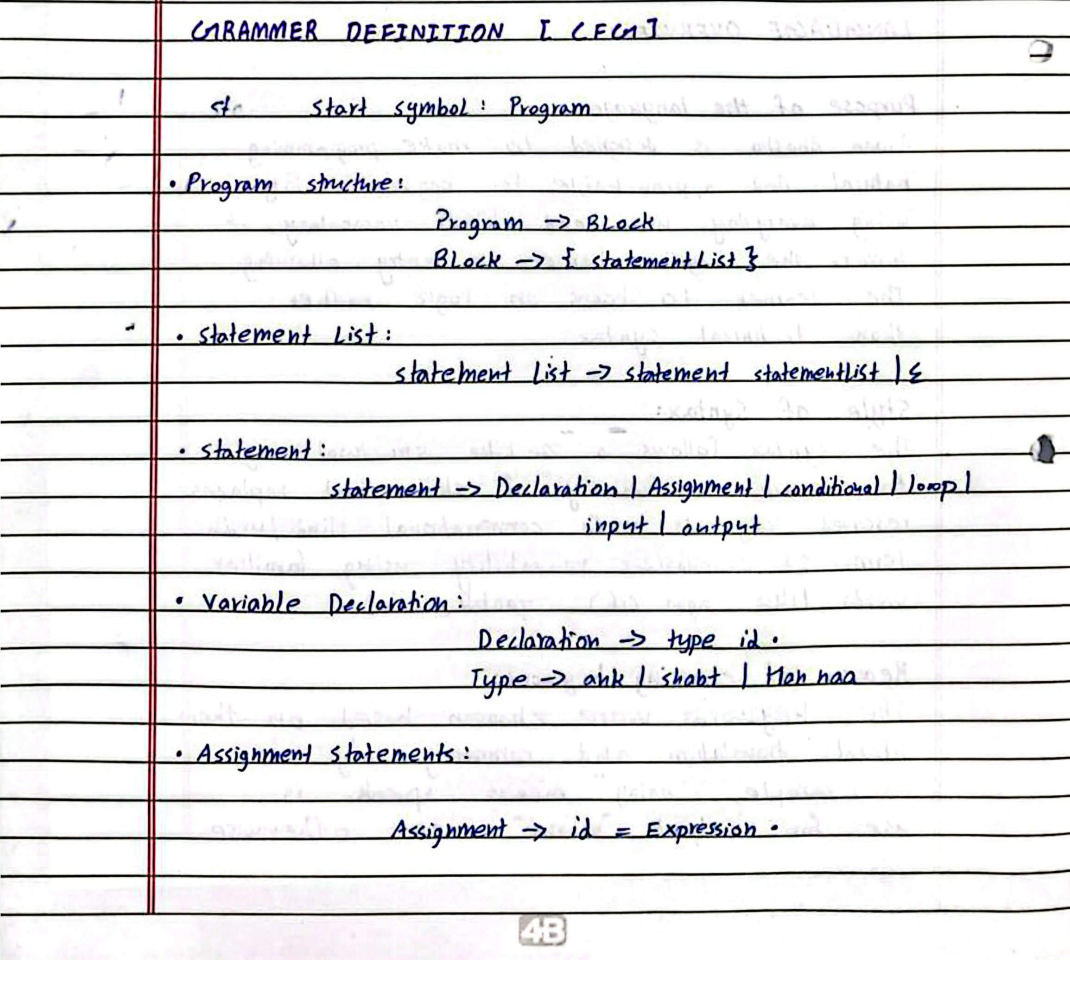
**Explanation of Keywords:**

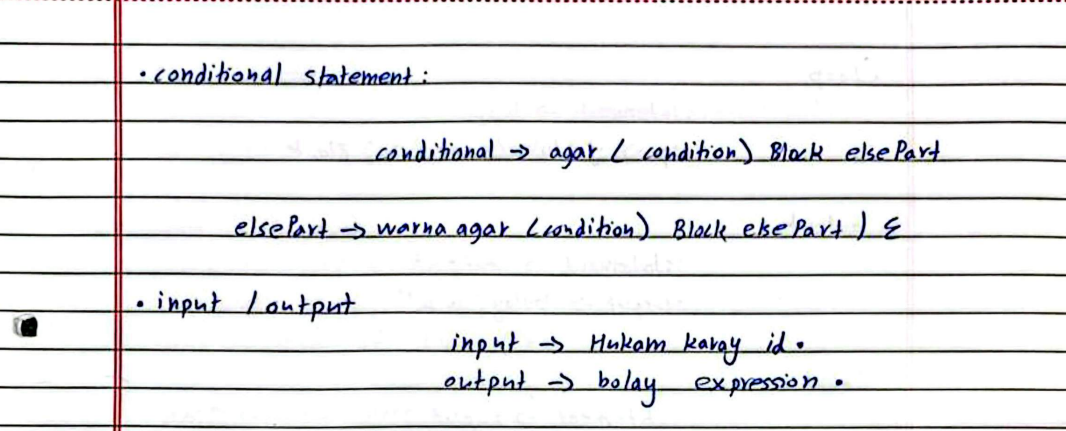
1. **agar** – This is a *Urdu* word used instead of **if**. It is a common everyday term meaning “if,” making it simple and natural for people to understand.
2. **warna** – A *Urdu* word used in place of **else**. It commonly means “otherwise” or “alternative,” which makes it easy for people to relate to.
3. **bolay** – A *Urdu* word used instead of **print**. It literally means “to speak” or “to say.” When someone wants to display or say something, “bolay” feels intuitive and easy to understand.
4. **warna agar** – A *Urdu* phrase used for **else if**. It naturally conveys the idea of “otherwise if” in spoken language.
5. **Hukam karay** – A *Urdu* expression used in place of **input**. It is a polite way of saying “please command” or “please ask,” which makes it respectful and easy to grasp.
6. **thehr jao** – A *Urdu* phrase meaning **stop**. It is used when one wants to pause or break a process, similar to the programming term *break*.
7. **ank** – A *Hindi* word used for **int** (numbers). In traditional Hindi, *ank* means “number,” so it feels familiar and easy to understand.
8. **shabt** – A *Hindi* word used for **string**. It means “word(s),” and is often heard in daily speech such as “kuch shabt bolay” (say a few words).
9. **jabtuk** – A *Urdu* word used for **while**. It translates to “until” or “as long as,” which perfectly matches the meaning of a while-loop condition.
10. **Han Naa** – A *Hindi* phrase meaning **yes or no**. It is frequently used in conversation, making it a simple and relatable choice for a boolean data type.
11. **Shudh** – A *Hindi* word meaning **pure** or **true**. It is used to represent the logical value *true* in an easily understood form.
12. **jhoot** – A *Urdu* word meaning **false** or “a lie.” It represents the opposite of *Shudh* (true) in a very natural way.
13. **lotaao** – A *Hindi* word meaning **return** or “give back.” It fits perfectly for returning a value in programming.
14. **challay** – A *Urdu* word used for **for loop**. It gives the sense of repetition or movement, which aligns with loop behavior.
15. **karo** – A *Urdu* word used for **do**. It is simple, direct, and naturally expresses the idea of performing an action.

**Operators and Punctuations:**

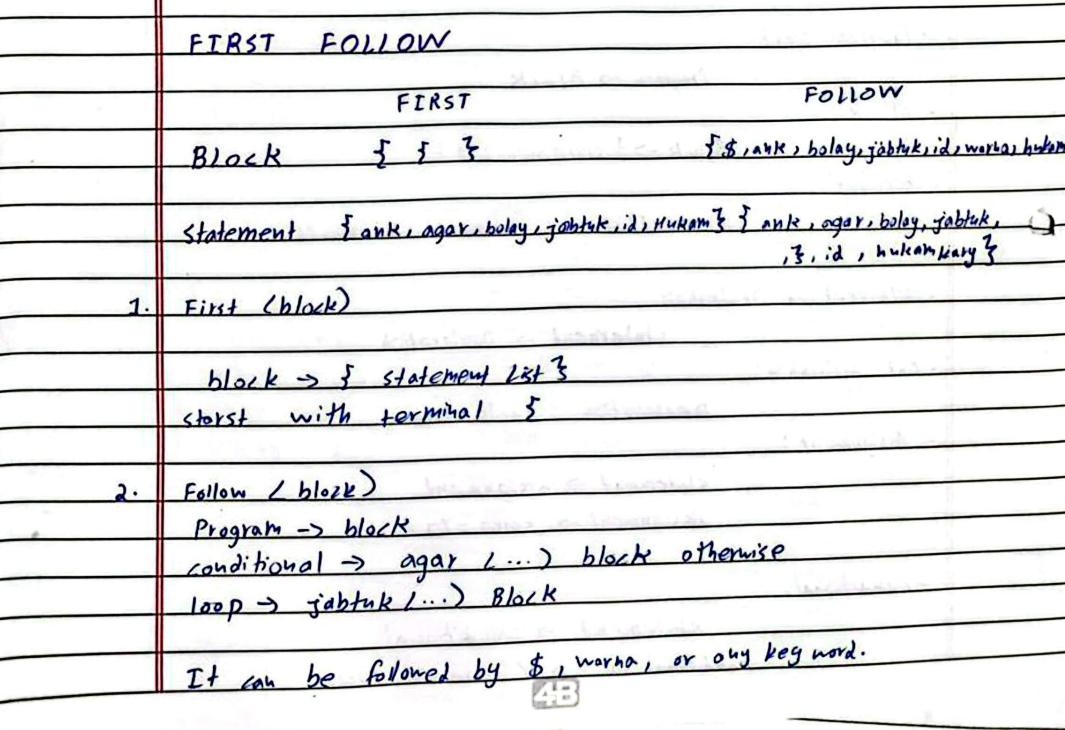
| **Category** | **Symbol** | **Description** |
| --- | --- | --- |
| **operator** | “ - ” | subtraction |
| **operator** | “ + ” | addition |
| **operator** | “ / ” | divide |
| **operator** | “ \* ” | multiply |
| **operator** | “ == ” | comparison |
| **operator** | “ > ” | greater than |
| **operator** | “ < ” | less than |
| **punctuation** | . | end of statement |
| **punctuation** | , | seperator |
| **punctuation** | ( ) | group/function call |
| **punctuation** | { } | block of code |
| **punctuation** | || | comment |

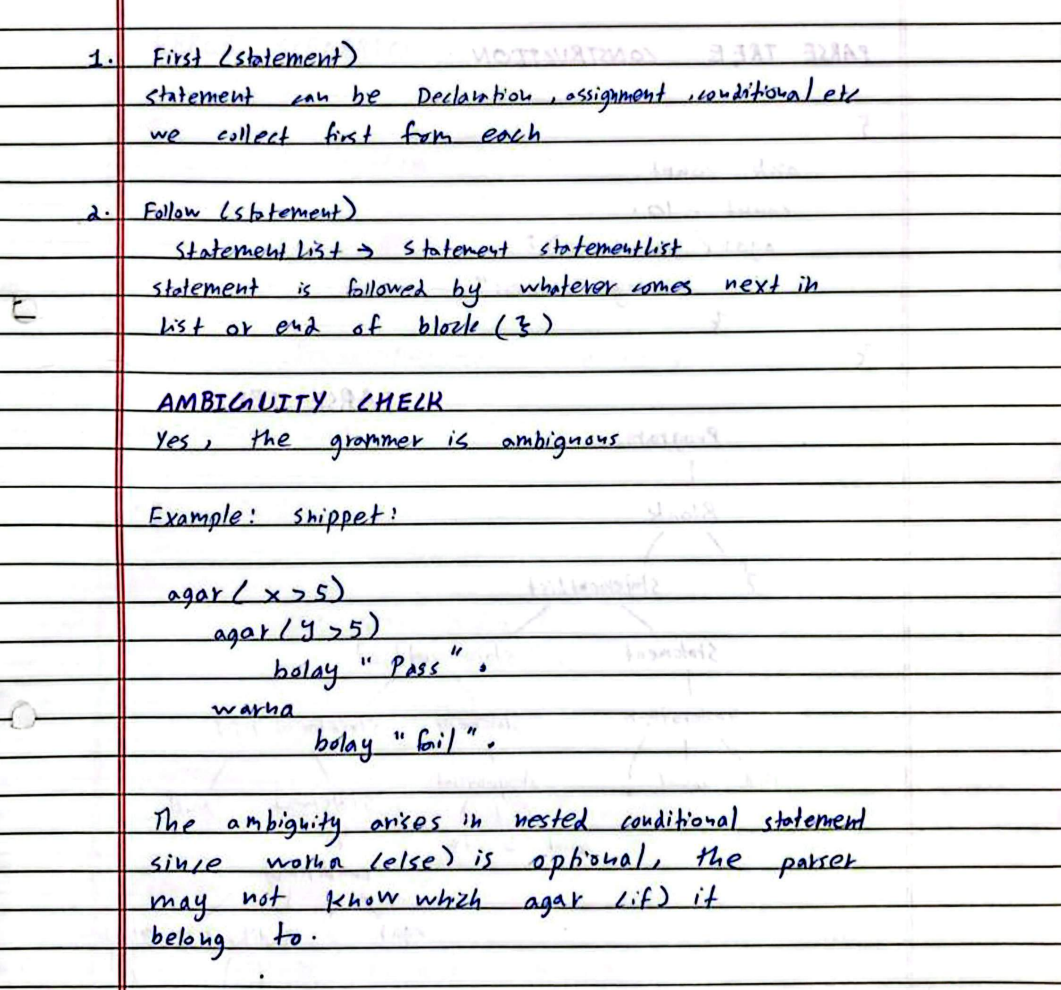
**Context Free Grammer:**

****

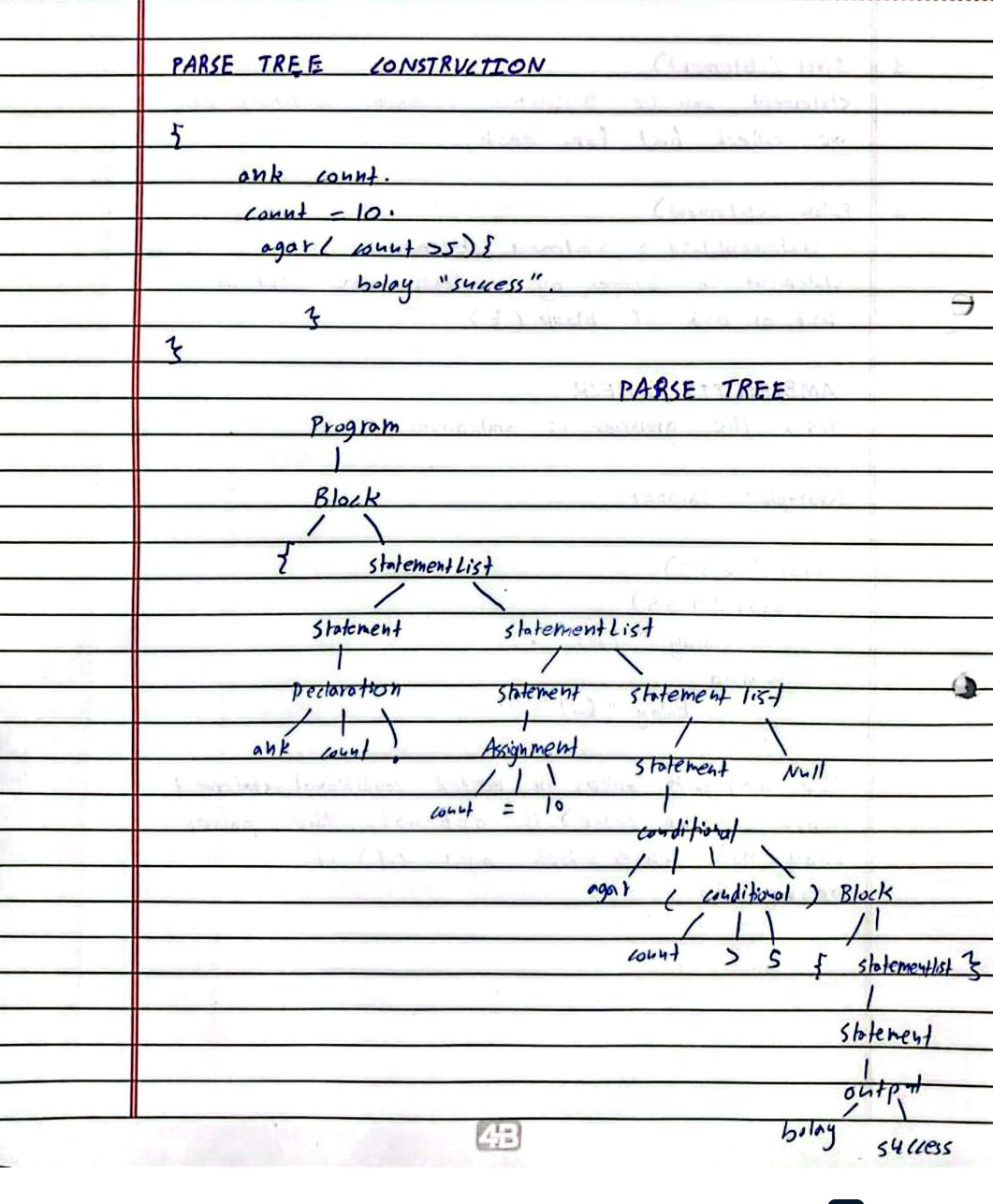


**First And Follow:**

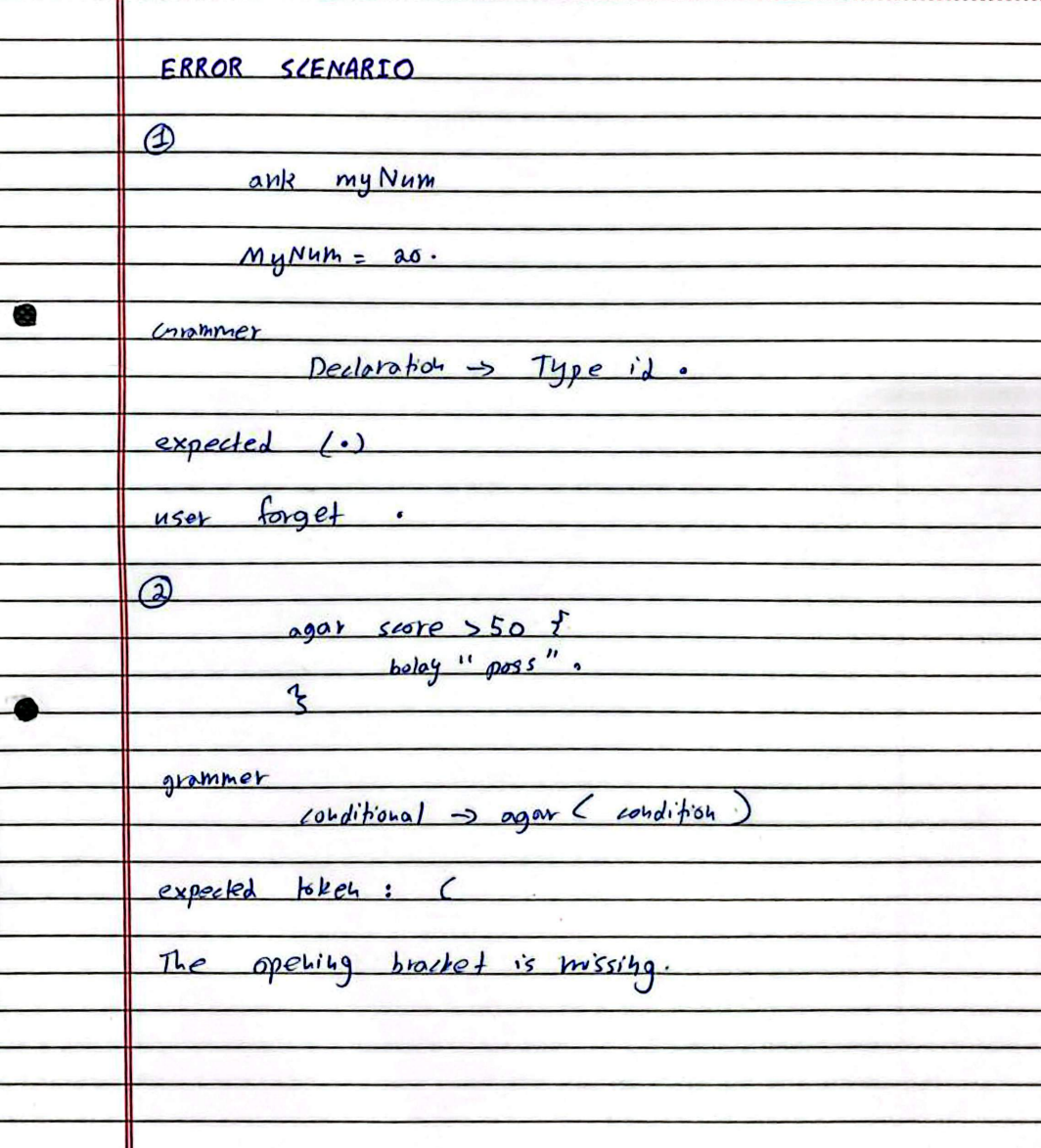
****

****

**Parse Tree:**

****

**Error Handling:**

****

**Phase 01 to Phase 02 Integration Explanation**

* **Token Identification (Scanner):** The Phase 01 lexical analyzer, implemented using Flex (scanner.l), scans the source code and identifies valid lexemes such as keywords (e.g., agar, bolay), operators, and identifiers, converting them into tokens.
* **Token Transmission (Scanner to Parser):** These tokens are passed to the Phase 02 parser, implemented using Bison (parser.y). The scanner communicates with the parser by including the header file (parser.tab.h) generated by Bison, ensuring both components use identical integer IDs for tokens.
* **Token Consistency:** The %token declarations in parser.y (e.g., AGAR, ANK, CHALLAY) exactly match the return values defined in scanner.l. This ensures a seamless handshake between the lexical and syntax analysis phases.
* **Syntax Validation:** The parser receives the stream of tokens and verifies them against the Context-Free Grammar (CFG) rules defined for Asaan Bhasha. It checks the structural correctness of the code (e.g., ensuring a JABTUK loop has the correct parentheses and block structure) without performing semantic analysis (like type checking).
* **Outcome:** If the input program strictly follows the grammar rules, the parser prints "Syntax Analysis Successful". If a violation occurs, the yyerror function is triggered to report the specific line number and error nature.

**REPOSITORY LINK:**

**https://github.com/hassan0981/AsaanBhasha**