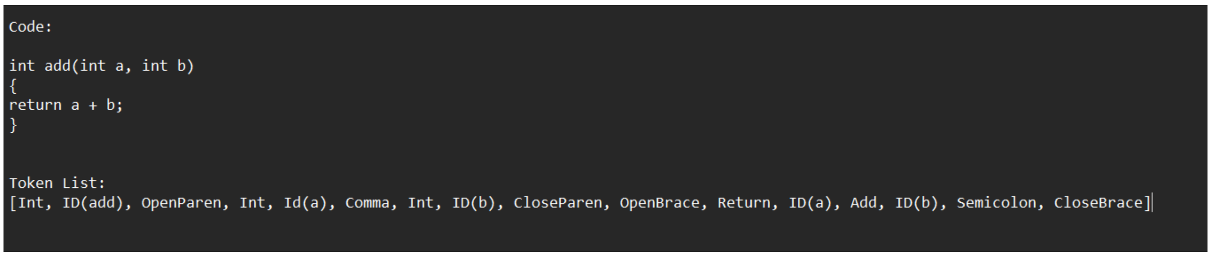
Sultan Alhosani 100053471

Key Algorithms

Lexical Analysis

At this stage is whereby the full source code is converted into various tokens such as keywords, identifiers, numeric values, and operators. The next() function in C4 reads the input and parses the characters into meaningful units, filtering out comments and spaces.

For example, in the function generates a token list.



```
Code:

int add(int a, int b)
{
return a + b;
}


Token List:
[Int, ID(add), OpenParen, Int, Id(a), Comma, Int, ID(b), CloseParen, OpenBrace, Return, ID(a), Add, ID(b), Semicolon, CloseBrace]
```

Moreover, the C4 lexical analyser uses hashing to optimize token lookup and a quicker identifier retrieval.

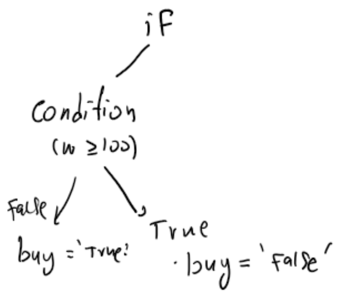
Parsing

It defines or organizes all tokens into an Abstract Syntax Tree (AST), letting the compiler know the logical relationship between this state. C4 implements recursive descent parsing to support constructs such as statements conditionals, loops, and expressions while respecting operator precedence.

For example, given this code:

if (w >=100) { buy= ‘False’; } else { buy= ‘True’; }

The code (C4) Generates this tree:



Sultan Alhosani 100053471

Virtual Machine

C4 interprets code into bytecode that is executed by a virtual machine (VM). Instead of generating CPU-targeted assembly such as in other compilers, the VM executes stack-based bytecode instructions, such as ADD, MUL, JMP

For instance, the function:

int multiply(int a, int b) { return a * b; }

becomes:

ENT 2 which is used to allocate stack space

LI a which is used to load a onto stack

LI b   which is used to load b onto stack

MUL   which is used to multiply top values

LEV   which is used to return result

Memory Management

C4 uses stack and heap allocation to perform memory management. Local variables and functions call frames are kept in the stack while dynamic allocations in the heap are made using the functions malloc and free.

For example:

- **Stack allocation (automatic memory management)**
- **void function() { int x = 5; }**

- **Heap allocation (manual memory management)**
- **void allocate() { int *ptr = (int *) malloc(10 * sizeof(int)); free(ptr); }**