# Team 2 Report

| **Name** | **Sec** | **B.N** | **Code** |
|---|---|---|---|
| Yousef Osama Mohamed | 2 | 32 | 9211386 |
| Nesma Abdelkader | 2 | 28 | 9211292 |
| Sara Gamal Gerges | 1 | 20 | 9210455 |
| Eman Ibraheem | 1 | 14 | 9210265 |

## Project Overview

This project involves the development of a basic compiler for a custom programming language, utilizing Lex and Yacc. The compiler is designed to handle fundamental programming constructs, including:

- Variable declarations
- Assignment operations
- Conditional statements (e.g., if-else)
- Looping constructs (e.g., while, for)
- Print/output statements
- Support for constants and basic arithmetic operations

---

## Tools and Technologies

- **Lex**: Employed for lexical analysis, transforming source code into a sequence of tokens.
- **Yacc**: Used for syntax analysis, constructing a parser based on the defined grammar.
- **C++**: The primary programming language used to implement the compiler's functionality

## Tokens

| Token | Example Usage | Description |
|---|---|---|
| - | a - b | Subtraction operator |
| ( | print(a) | Start of parentheses |
| ) | print(a) | End of parentheses |
| < | if (a < b) | Less than comparison |
| > | if (a > b) | Greater than comparison |
| = | a = 5 | Assignment operator |
| + | a + b | Addition operator |
| * | a * b | Multiplication operator |
| / | a / b | Division operator |
| % | a % b | Modulus (remainder) |
| { | if (a) {} | Start of a block |
| } | if (a) {} | End of a block |
| : | case 1: | Used in switch-case |
| ; | a = 5; | Statement terminator |

| | | |
|---|---|---|
| , | print(a, b) | Separator between arguments or items |
| const | const int x = 5; | Declare a constant |
| int | int x; | Declare an integer |
| float | float pi = 3.14; | Declare a floating-point number |
| string | string name = "Alice"; | Declare a string |
| bool | bool isTrue = true; | Declare a boolean |
| void | Def void func() {} | Declare a function with no return value |
| print | print("Hello"); | Print output |
| return | return 0; | Return from a function |
| for | for (i = 0; i < 10; i++) {} | For loop |
| do | do { } until (x < 5); | Do-while loop |
| while | while (i < 10) {} | While loop |
| until | do { } until (x == 5); | Loop until condition is met |
| if | if (x > 0) {} | Conditional if statement |
| else | else {} | Else block after if |

| | | |
|---|---|---|
| switch | switch (x) {} | Switch between multiple cases |
| case | case 1: | Define a case inside switch |
| def | def int  add(a, b) {} | Function definition |
| default | default: | Default case for switch |
| break | break; | Exit a loop or switch |
| continue | continue; | Skip to next loop iteration |
| and | if (a > 0 and b < 5) | Logical AND |
| or | if (a > 0 or b < 5) | Logical OR |
| not | if (not a) | Logical NOT |
| true | bool flag = true; | Boolean true value |
| false | bool flag = false; | Boolean false value |
| >= | if (a >= b) | Greater than or equal |
| <= | if (a <= b) | Less than or equal |
| == | if (a == b) | Equality check |
| != | if (a != b) | Not equal check |

| | | |
|---|---|---|
| ++ | i++ | Increment a variable |
| -- | i-- | Decrement a variable |
| INTEGER | 123 | A whole number |
| FLOAT | 3.14 | A decimal number |
| STRING | "Hello, World!" | A string of text |
| VARIABLE | counter | A variable name |
| # comment | # This is a comment | Ignored during execution |
| Whitespace | spaces, tabs | Ignored by lexer |
| Newline (\n) | (line break) | Advances line count |
| Unknown characters | @, ~, etc. | Cause syntax error |

## quadruples

**1. push (type) value**

- Description: Pushes a constant value of a specific type onto the stack.

**2. push identifier**

- Description: Pushes the value of an identifier (variable) onto the stack.

**3.pop identifier**

- Description: Pops a value from the stack and stores it in the specified identifier (variable).

**4. print (type)**

- Description: Pops a value of the specified type from the stack and prints it.

**5. Call function_name**
- Description: Calls a function with the specified name, using arguments previously pushed onto the stack.

**6. proc function_name**
- Description: Marks the beginning of a function definition.

**7. endproc**
- Description: Marks the end of a function definition.

**8. ret**
- Description: Returns a value from a function by popping it from the stack.

**9.jmp label**
- Description: Unconditionally jumps to the specified label.

**10.jz label**
- Description: Jumps to the specified label if the top stack value (a boolean) is zero (false).

**11.jnz label**
- Description: Jumps to the specified label if the top stack value (a boolean) is non-zero (true).

**12.compEQ**
- Description: Compares the top two stack values for equality, pushing a boolean result.

**13.add**
- Description: Pops two values, adds them, and pushes the result.

**14.sub**
- Description: Pops two values, subtracts the second from the first, and pushes the result.

**15.mul**
- Description: Pops two values, multiplies them, and pushes the result.

**16.div**
- Description: Pops two values, divides the first by the second, and pushes the result.

**17.mod**
- Description: Pops two values, computes the modulus (remainder), and pushes the result.

**18.lt**
- Description: Pops two values, compares if the first is less than the second, and pushes a boolean result.

**19.gt**
- Description: Pops two values, compares if the first is greater than the second, and pushes a boolean result.

**20.ge**
- Description: Pops two values, compares if the first is greater than or equal to the second, and pushes a boolean result.

**21.le**
- Description: Pops two values, compares if the first is less than or equal to the second, and pushes a boolean result.

**22.eq**
- Description: Pops two values, compares if they are equal, and pushes a boolean result.

**23. ne**

- Description: Pops two values, compares if they are not equal, and pushes a boolean result.

**24. and**

- Description: Pops two boolean values, computes their logical AND, and pushes the result.

**25. or**

- Description: Pops two boolean values, computes their logical OR, and pushes the result.

**26. not**

- Description: Pops a boolean value, computes its logical NOT, and pushes the result.

**27. neg**

- Description: Pops a numeric value, computes its negation (unary minus), and pushes the result.

**28.pre inc**

- Description: Increments the value of a variable before using it and pushes the new value.

**29. Pre dec**

- Description: Decrements the value of a variable before using it and pushes the new value.

**30. post inc**

- Description: Pushes the current value of a variable and then increments it.

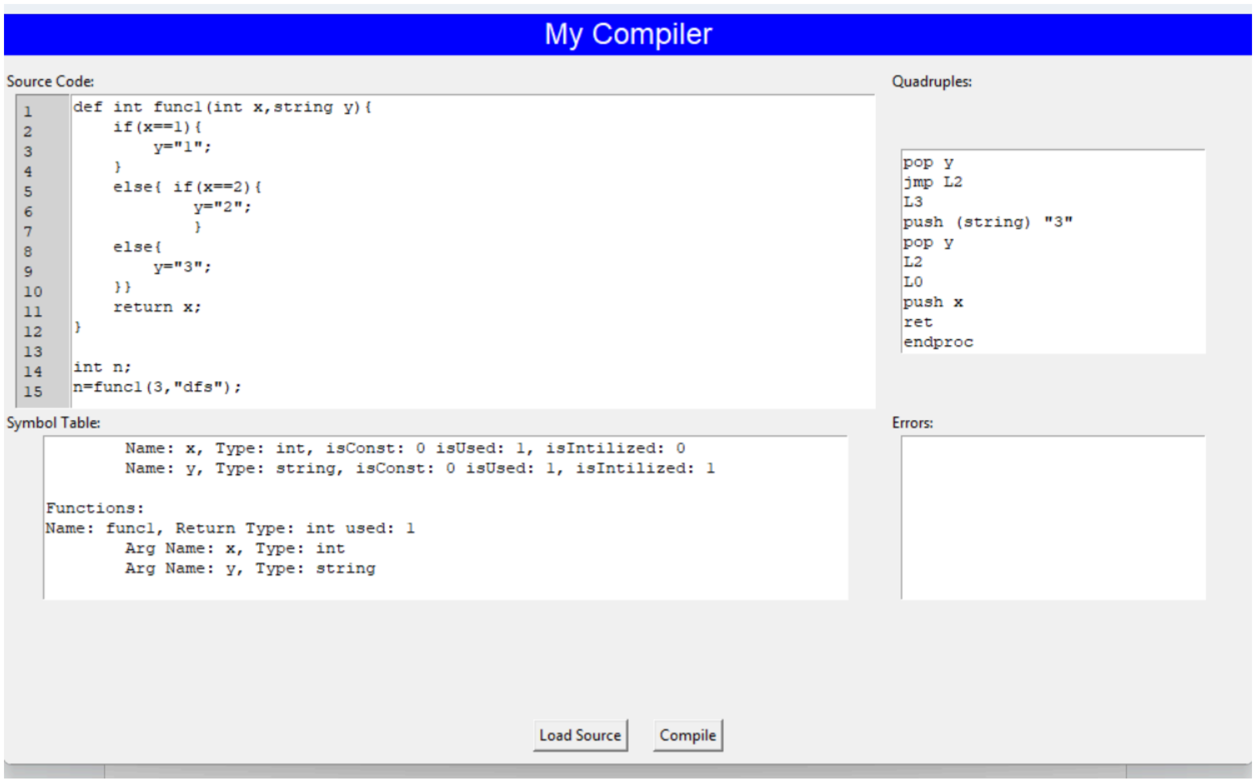
**31. post dec**

- Description: Pushes the current value of a variable and then decrements it.

**32. Convert**

- Cast from int to float & from float to int

## EXTRA



```
def int func1(int x,string y){
    if(x==1){
        y="1";
    }
    else{ if(x==2){
            y="2";
            }
    else{
        y="3";
    }}
    return x;
}

int n;
n=func1(3,"dfs");
```

```
pop y
jmp L2
L3
push (string) "3"
pop y
L2
L0
push x
ret
endproc
```

```
        Name: x, Type: int, isConst: 0 isUsed: 1, isIntilized: 0
        Name: y, Type: string, isConst: 0 isUsed: 1, isIntilized: 1

Functions:
Name: func1, Return Type: int used: 1
        Arg Name: x, Type: int
        Arg Name: y, Type: string
```

1. Handle casting from int to float & from float to int
2. Semantic error: argument function mismatch
3. Function Argument Type Mismatch
4. Modification of a Constant Variable
5. Invalid Unary Operation (eg, applying not to a float variable)
6. Using a non-boolean expression as the condition