Compilers Assignment 1

- Group 6
- Reuben Devanesan, 19110059
- Kamal Vaishnav, 20110089
- Preetam Chhimpa, 20110145
- Rishab Jain, 20110164

GullyLang: Introduction

As part of our first assignment, we have constructed a simple programming language **"GullyLang"**. We have taken inspiration from **Bhailang**, a dynamically typed toy programming language that took social media by storm in 2022. **GullyLang** (a tribute to the popular term *Gully Gang*) incorporates commonly used Hindi words and references to popular Indian memes and slangs, creating a unique and culturally resonant programming experience.

GullyLang: Documentation

1) General

- The syntax of comments are the same as in other programming languages. ***//*** for single line comments and ***/* */*** for multi-line comments.
- GullyLang is case sensitive. A semicolon ***;*** is used to indicate the end of statements.
- A pair of curly braces ***{}*** is used to delimit a block statement.
- To print anything to the console, the keyword ***bole toh*** is used, a reference to a popular slang in Mumbai.

```
bole toh "Namaste Duniya!" ;
```

2) Variables & Data Types

- Keywords: *maanle, kuch nhi, genuine, condemn*
- Variables can be declared using the ***maanle*** keyword, which means to assume in Hindi.
- Null values can be denoted using ***kuch nhi*** keyword and booleans values can be denoted using ***genuine*** (Truth value) and ***condemn*** (False value). Both boolean values are based on an inside joke of our friend group :)
- Numbers, strings, lists and maps are just like other programming languages.

```
maanle name = "Gully Coder";
maanle sankhya = 100;
maanle khali = kuch nhi; // Null Value
maanle sach = genuine; // Truth Value
maanle jhoot = condemn; // False Value
maanle list_example = [1, 2, 3];
```

3) Operators

- GullyLang has operators just like other programming languages.
- Operators in order of precedence (not an exhaustive list):

Operator	Description	Usage
()	Grouping	(x)
*, /, %	Multiplication, Division, Remainder	x * y, x / y, x % y
+, -	Addition, Subtraction	x + y, x - y
>, >=, <, <=	Relational Operators	x > y, x >= y, x < y, x <= y
==, !=	Equality Operators	x == y, x != y
=	Assignment	x = y

4) Conditionals

- Keywords: *ya toh, nahi toh, varna*
- GullyLang supports the if-else-if ladder construct , ***ya toh*** block will execute if condition is ***genuine***, otherwise one of the subsequently added ***nahi toh*** blocks will execute if their respective condition is ***genuine***, and the ***varna*** block will eventually execute if all of the above conditions are ***condemn***.

```
ya toh (condition 1) {
    bole toh "Ye toh sach hai!";
} nahi toh (condition 2) {
    bole toh "Ye toh sach nikla!";
} varna {
    bole toh "Kuch bhi nahi pata!";
}
```

5) Loops & Iterations

- Keywords: *jab tak, jitni baar, rukja, chalte re*
- Statements inside ***jab tak*** blocks are executed as long as a specified condition evaluates to ***genuine***. If the condition becomes ***condemn***, the loop stops executing and control passes to the statement following the loop.
- Use ***rukja*** to break the loop and ***chalte re*** to continue within the loop.
- Gully lang also provides another loop with syntax similar to for loops in JavaScript.. Use ***jitni baar*** to execute the loop.

```
maanle counter = 1;
jab tak (counter <= 5) {
    bole toh counter;
    counter = counter + 1;
    ya toh (condition 1) {
        rukja; // to break the loop
    } varna {
        chalte re; // to continue in the loop
    }
}
```

6) Functions

- Keywords: *kaam kar, nikal*
- GullyLang also provides support for blocks of code that run only when code i.e functions. Functions can be defined using the ***kaam kar*** block.
- Use the ***nikal*** keyword to exit from a function or to return particular values to the called statement.
- A function definition consists of the ***kaam kar*** keyword, followed by:
  - The name of the function.
  - List of parameters to the function, enclosed in parentheses, separated by commas
  - GullyLang statements that define the function, enclosed in curly braces, *{ /* … */ }*

```
kaam kar greeting (naam) {
    bole toh "Namaste, " + naam;
    nikal;
}
// calling the function
greeting ("GullyCoder");
```

7) Error Handling

- Keywords: *chalake dekh, aayein baigan*
- GullyLang also provides support for error handling. The ***chalake dekh*** statement allows you to define a block of code to be tested for errors while it is being executed. The ***aayein baigan*** statement allows you to define a block of code to be executed, if an error occurs in the ***chalake dekh*** block.

```
chalake dekh {
    // testing code here
} aayein baigan {
    bole toh "Kuch toh gadbad hai!";
}
```