As part of this course, we will be implementing a simple interpreter for a made-up programming language that, for lack of a better name, we’ll call MyPL.

**Instructions:** For this assignment, you must write five non-trivial MyPL example programs. Across the 5 examples you should use each construct at least once. For each program describe what it does. Turn in each of your programs, typed, with the corresponding descriptions of each. The basic constructs of the programming language are described informally below.

**Informal Description of MyPL:** The made-up programming language we’ll be implementing in this course is a C-like procedural language. The language supports basic assignment statements, arithmetic expressions, conditionals, loops, lists, and basic input/output. All variables are implicitly typed (i.e., the types of variables are inferred from their values). We’ll discuss how typing works in MyPL later in the semester. Programs consist of a sequence of statements given within a single file. The language constructs supported by MyPL are described in more detail below.

1. **Primitive Data Types.** MyPL supports integer, string (denoted by double quotes, e.g., "Hello World!"), and Boolean values (true and false).

2. **List Types.** MyPL also supports Python-like lists. Lists are denoted by square brackets, e.g., “[1,2,3]”. List variables can be indexed, e.g., “xs[0]” to get the value in the first list position or to assign a value into the first list position.

3. **Assignment Statements.** An assignment statement takes the form “var = expr;” where var is a valid identifier (a letter followed by zero or more letters, digits, or underscores) and expr is a valid expression. Assignment statements bind the variable to the value that results from evaluating the expression. Assignment statements must end in a semicolon.

4. **Output Statements.** An output statement takes the form “print(expr);” or “println(expr);”. Print sends the value that results from evaluating the expression to standard output (the terminal). A println statement adds a newline to the result whereas a print statement does not. Print statements must end in a semicolon.

5. **Math Operators.** The typical math operators +, -, *, /, and % (modulus) are supported. Note that we only support integer division (e.g., the expression “5/2” evaluates to 2).

6. **Relational Operators.** The relational operators ==, <, >, <=, >=, and != are supported.

7. **Boolean Connectives.** The Boolean connectives and, or, and not are supported.

8. **Input Expressions.** User input is obtained through “readint(msg)” and “readstr(msg)” expressions, where “msg” is a string value. For example, “ans = readint("Enter an int: ")” prompts the user using the message “Enter an int: ”, and then after the user enters an integer value and hits “Enter”, the value is stored in the variable ans. A readint expression
assumes an integer value is entered, whereas a `readstr` expression treats the input as a string value. Both `readint` and `readstr` expressions can occur anywhere an integer or string value would be used, respectively. For example, “`println(5 + readint("Enter an int: "));`” is a valid statement in MyPL.

9. **While Statements.** A while statement takes the form “`while bool-expr do stmts end`”, where `bool-expr` is a Boolean expression and `stmts` is a list of statements.

10. **Conditional Statements.** A condition statement takes the form “`if bool-expr then stmts elseif bool-expr then stmts else stmts end`”. A conditional statement can have zero or more `elseif` clauses and zero or one `else` clause. A conditional statement always ends with an “`end`” reserved word. Note that `elseif` is a distinct reserved word and should be used instead of an `else` followed by an `if`.

11. **Comments.** Single-line comments are denoted by the “`#`” symbol. That is, everything on a line after a “`#`” symbol is ignored.

The following are some simple examples of statements in MyPL.

```plaintext
# obligatory hello world program
println("Hello world!");

# simple conditional statement
x = readint("Enter an int: ");
y = readint("Enter an int: ");
if x > y then
    println("The first int was bigger than the second!");
elseif y > x then
    println("The second int was bigger than the first!");
else
    println("You entered the same value twice!");
end

# simple while statement
z = readint("Enter an int: ");
i = 0;
while z > 2 do
    z = z / 2;
i = i + 1;
end
print("z = ");
print(z);
print(" , i = ");
println(i);
```