ArnoldC Interpreter

Specifications

The minimum requirement for the project is listed below.

File naming and Formatting

- ArnoldC source files should have the `.arnoldc` file extension.
- Execution of ArnoldC programs start inside its *main function.* The main function in ArnoldC is enclosed by the keywords `IT'S SHOWTIME` (beginning delimiter) and `YOU HAVE BEEN TERMINATED` (ending delimiter).
- Function declarations can be placed before or after the main function.

✔	`LISTEN TO ME VERY CAREFULLY hi`
	`  TALK TO THE HAND "hi"`
	`HASTA LA VISTA, BABY`
	
	`IT'S SHOWTIME`
	`  DO IT NOW hi`
	`YOU HAVE BEEN TERMINATED`

❌	`HEY CHRISTMAS TREE x`
	
	`IT'S SHOWTIME`
	`  TALK TO THE HAND "Hi"`
	`YOU HAVE BEEN TERMINATED`

✔	`IT'S SHOWTIME`
	`  DO IT NOW hi`
	`YOU HAVE BEEN TERMINATED`
	
	`LISTEN TO ME VERY CAREFULLY hi`
	`  TALK TO THE HAND "hi"`
	`HASTA LA VISTA, BABY`

Spacing/Whitespaces

- You may assume that one line contains one statement only. Each statement is delimited by the new line.
- You may assume that there is only one whitespace between keywords. Indentation is irrelevant.

✔	`HEY CHRISTMAS TREE varName`
❌	`HEY  CHRISTMAS  TREE varName`
✔	`  HEY CHRISTMAS TREE varName`

- Spaces inside a `string` literal should be retained.

`"Spaces  between"`	*should **NOT** become "Spaces between"*

Comments

- Currently, there is no support for comments in ArnoldC.

Variables

- Variables should be declared inside functions (either in the main or in the user-defined functions).
- Variable names must start with a letter, followed by any combination of letters, numbers, and underscores. No spaces, dashes, or other special symbols are allowed to be part of the variable name.
- Variable declaration is done using the keyword `HEY CHRISTMAS TREE`.
- Variable initialization after declaration is **required** and is done using the `YOU SET US UP` keyword.
- The data type of a variable is an **integer.**
- The variable can be initialized with some **integer** value or a **macro**.

```
HEY CHRISTMAS TREE var1
YOU SET US UP @NO PROBLEMO
HEY CHRISTMAS TREE var2
YOU SET US UP @I LIED
HEY CHRISTMAS TREE var3
YOU SET US UP 123456
```

- Values of variables can be reassigned using the `GET TO THE CHOPPER`, `HERE IS MY INVITATION`, and `ENOUGH TALK` keywords.

```
HEY CHRISTMAS TREE var1
YOU SET US UP 10

GET TO THE CHOPPER var1
HERE IS MY INVITATION 99
ENOUGH TALK
```

- If the value that will be assigned to a variable is a result of an arithmetic operation, the arithmetic operations must be inserted between the `HERE IS MY INVITATION`, and `ENOUGH TALK` keywords.

```
HEY CHRISTMAS TREE var1
YOU SET US UP 10

GET TO THE CHOPPER var1
HERE IS MY INVITATION 99
GET DOWN 9
ENOUGH TALK
```

Data Types

- ArnoldC only has two data types: **integer** and **strings**.
- These data types can come in form of literals. Integers, on the other hand, can also come in the form of a variable.
- Variables cannot hold strings.
- There is **no boolean data type** in ArnoldC. Any non-zero integer is considered **true**; zero (0) is considered **false**. True expressions result to the value one (1) while false expressions result to zero (0).

Operations

- Since ArnoldC only has **integers** as data types (and **strings** are mainly used for printing), operations are only arithmetic.
- Execution of operations can only be done together with assignment statements.

ARITHMETIC/MATHEMATICAL OPERATIONS

- Below are the arithmetic operations:

```
GET TO THE CHOPPER myVar
HERE IS MY INVITATION 2
GET UP 3                          ADDITION
GET DOWN 4                        SUBTRACTION
YOU'RE FIRED 5                    MULTIPLICATION
HE HAD TO SPLIT 6                 DIVISION
ENOUGH TALK
```

- `String`s should not be used in arithmetic operations.
- Operands of arithmetic operations can be variables and/or literals.

LOGICAL OPERATIONS

- Below are the logical operations:

```
GET TO THE CHOPPER myVar
HERE IS MY INVITATION @I LIED
YOU ARE NOT YOU YOU ARE ME a      EQUAL TO
LET OFF SOME STEAM BENNET b       GREATER THAN
CONSIDER THAT A DIVORCE c         OR
KNOCK KNOCK d                     AND
ENOUGH TALK
```

OUTPUT

- Printing can be done using the `TALK TO THE HAND` keyword. It takes only one parameter: a variable or a string literal.

```
TALK TO THE HAND "Value of x: "
TALK TO THE HAND x
```

INPUT

- Getting an input is done and assigning the input to a variable is done using the `GET YOUR ASS TO MARS`, `DO IT NOW`, and `I WANT TO ASK YOU A BUNCH OF QUESTIONS AND I WANT TO HAVE THEM ANSWERED IMMEDIATELY` keywords.

```
GET YOUR ASS TO MARS var1         assign output to var1
DO IT NOW                         call the input function
I WANT TO ASK YOU A BUNCH OF QUESTIONS AND I WANT TO HAVE THEM
   ANSWERED IMMEDIATELY           this is the input function
```

Statements

EXPRESSION STATEMENTS

- The result of an expression should always be assigned to a variable using the `GET TO THE CHOPPER`, `HERE IS MY INVITATION`, and `ENOUGH TALK` keywords.

FLOW-CONTROL STATEMENTS

IF-THEN STATEMENTS

- ArnoldC has If-Then uses the keywords `BECAUSE I'M GOING TO SAY PLEASE` and `YOU HAVE NO RESPECT FOR LOGIC`.

```
HEY CHRISTMAS TREE x
YOU SET US UP @I LIED

BECAUSE I'M GOING TO SAY PLEASE x
   TALK TO THE HAND "true"
BULLSHIT
   TALK TO THE HAND "false"
YOU HAVE NO RESPECT FOR LOGIC
```

- Indentation is irrelevant.
- If there is no ELSE clause, the `BULLSHIT` keyword may not be used.

ITERATION

- ArnoldC implements a while-loop using the keywords `STICK AROUND` and `CHILL`.

```
HEY CHRISTMAS TREE isFive
YOU SET US UP 5

STICK AROUND isFive
   TALK TO THE HAND isFive
   GET TO THE CHOPPER isFive
   HERE IS MY INVITATION isFive
   GET DOWN 1
   ENOUGH TALK
CHILL
```

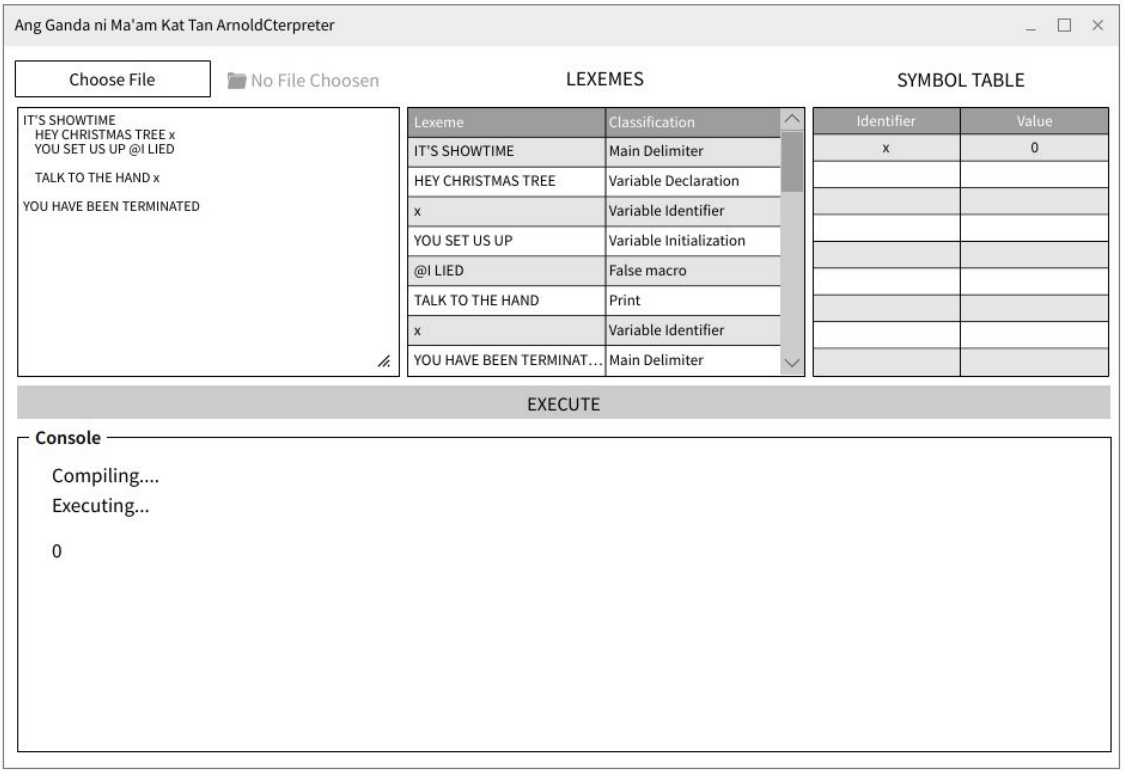
Extra Credit

Implementing anything that is not specified in this document will give you bonus points. The number of bonus points depends on the level of difficulty of the feature you implement. Possible bonus features are:

Void functions w/o parameters	Non-void functions w/o parameters	Loop-Nesting
Void functions w/ parameters	Non-void functions w/ parameters	

Submission Format

- **You may use any programming language** that you want to create the interpreter.
- A **Graphical User Interface (GUI) is required** and should look similar to the diagram below:



- You are **NOT ALLOWED to use Flex/Lex or YACC/Bison or Parsing Expression Grammars (PEG) or ANYTHING SIMILAR**. You are required to implement your own lexical and syntax analyzer.
- Place all the files of your project in an archive file with the filename `<GroupName>_124Project.zip`. Your project should run if we extract and run it.
- Include a file called `contributors.txt` in the archive file. The text file should contain your lab section and full names.

```
B-0L
Zenith O. Arnejo
Maureen Lyndel C. Lauron
Clinton E. Poserio
Katherine Loren M. Tan
```

- Upload your project archive to your respective Google Classroom assignment posts.

Scoring

Since this is a group project, there will be a peer evaluation at the end of the semester. The peer evaluation score will be directly multiplied to your group's overall score in the project. You will evaluate each of your group mates AND yourself. Refer to the example below:

MEMBER	GROUP SCORE	PEER EVAL (Average)	FINAL PROJECT GRADE
Clinton	97.63%	100%	97.630
Kat		80%	78.104
Berna		60%	58.578

The peer evaluation has a big effect on your final project grade. Be mindful of the contributions you make for the group project and always put your best foot forward so that your group mates will want to give you the full 100% for their evaluation of your contributions, cooperativity, etc.

The breakdown for computing the project group score is:

SUB-UNIT		POINTS
Regular Expressions		20
Backus Naur Form		20
User Input/Output	Input	5
	Output String	3
	Output Variable	3
Variables	Declaration	5
	Initialization	5
Operations	Assignment	7
	Arithmetic	5
	Logical	5
If-Else	if-then	6
	if-else-then	6
Loops	While loop	10
	TOTAL	**100**