EZLang User Test

1.Commenting

Single-line comments are done with the # symbol. Multi-line comments are anything between a /\* and a \*/. Comments are ignored by the interpreter.

#This is a comment

Multi-line comment

/\*

This is also a comment.

\*/

White space is ignored.

2.DataTypes

integers: Whole counting numbers from 0

4567 #An integer

floats: Numbers with a fractional point. If a float is less than 1 it must start with a 0 not a dot.

0.4567 #A float

.4567 #Not a valid EZlang float

Chars: single characters and are sandwiched between two quotation marks

‘c’ #a char

Strings: multi-letter datatype. A string is sandwiched between two double quotation marks.

“String” #a string

Booleans: true or false. They cannot be used for variables and must be typed in lowercase.

true #the Boolean value true

false #the Boolean value false

null: Represents no value. Must be written in lowercase.

null # a null datatype

3. Data Structure

Arrays

In EZlang, arrays are heterogeneous so each index in an array can take in any valid EZlang datatype. Arrays are represented by a pair of square brackets, [ ], with values inside the brackets separated by a comma.

[1,2,3] # an array of integers

[1,’2’, “three”] # a heterogeneous array of mixed values

[] #an empty array

4. Ending a statement

Like Java, in EZlang every line of code must end with a semi-colon, ;. This turns the line of code into a statement that will be executed by EZlang’s interpreter.

5 #This is not a statement

5; #This is a statement

5. Variables

Declaring Variables

In EZlang variables must consist of one or a combination of lower case letters, uppercase letters, and/or an underscore, \_.

a\_variable; #this is a valid variable declaration

45hg!6; #This is not a valid variable declaration

Variable Assignment

After declaring a valid variable, an equal sign, =, can be used for assigning the variable to a value. Unlike Java, EZlang is dynamically typed so the type of the assigned value does not have to be explicitly written when declaring a variable. A variable can be assigned to any EZlang datatype or data structure. If a variable is not initialized or assigned a value it is null. An already declared variable can be reassigned to a different datatype.

a = 5; #declares a variable a and initializes it to the integer 5.

b= “String”; #declares a variable b and initializes it to a string

c= ‘c’; #declares a variable c and initializes it to a char

d = false; #declares a variable d and initializes it to the Boolean value false

array = [1,2,3]; #declares a variable array and initializes it to an array

x =5; #assigns integer 5 to x

x= “5”; #assigns string 5 to x

6. Print statement

A program can output a result to the user with a print statement. Any expression, or variable can be printed with a print statement.

print “printed”; #This will print printed

a =5; print a; #This will print 5

**Task 1:** Declare a variable, assign it to a value, then assign it to a different value in a comment, finally print the variable. Does the output match what you expect? Do this as many times as you like with different variables and assignments, once you’re satisfied move to the next part.

7. Reading values

You can assign a variable to the keyword read. The program will then ask you to input a value for the variable. Once a value is entered, it will be assigned to the variable.

x=read; #You will be prompted for to input a value for the variable x.

**Task 2:** Write an EZlang program that declares a variable, assigns it to read, and prints the value of the variable. Does the printed variable match what you entered?

8. Comparisons

Greater than(>), Less than(<), Greater than or equal(>=), Less than or equal (<=)

These operators are used to compare two operands. In EZlang they can only be used to compare numbers. They will compare the two numbers, and evaluate whether it is true or false and return the the result of the comparision as a Boolean.

5 >= 7; #This will evaluate to false

a=1< 10; print a; #a will print true

==, !=

These operators are used to check for equality and type between two operands, and will evaluate the operation and return the result as a boolean.

a=5==10; print a; this will print false since 5 is not equal to 10.

123 != “123”; #This will evaluate to true since integers and strings are not

the same type

and , or

These are identical to Java’s && and ||. They will compare the results of two comparisons. With and operations, if both comparisons are true, it evaluates to true otherwise it evaluates to false. With or operations, if any of the comparisons are true it evaluates to true, otherwise it evaluates to false.

print 5==5 and ‘c’==’c’; #Prints true since both are true

a= 5<=10 or 8!=10; print a; #Prints false since neither are true.

9. Unary Operators

!

The ! operator checks if an operand evaluates to false or not. If the operand is false, the whole expression evaluates to true and vice verse.

!(6==9); # This will evaluate to true since 6 is not equal to 9.

10. If-then-else statements

The above operators are useful in if-then-else statements.If-then-else is implemented in the same way as Java. If the expression in if is true code in the if brackets is executed otherwise it is skipped or the program executes code in an else bracket instead. If-then-else statements don’t have to have an else bracket.

If ( condition1 ) {

Code in here is executed if condition1 is true, otherwise it is skipped

} else {

Code in here is executed if condition1 is false

}

This is valid as well:

if ( condition ) {

code if condition is true

}

**Task 3**: Write a program that reads a user-inputted value for a variable x, if x is the integer 5 the program should print the String “You entered 5!”, if x is not the integer 5 it should print the String “You did not enter 5 :(“

11. Arithmetic Operations

+, -, \*, /, %

With the exception of chars, arithmetic operations are implemented in the same way as Java. -,\*, /, % only work with numbers. As well as numbers, + can be used for concatenating strings, concatenating 2 cars(which results in a String with 2 characters), and concatenating Strings and numbers with the result depending on the data type on the left hand operand. If you try to perform an arithmetic operation with invalid datatypes a runtime error will be produced. Like in Java, if you want a change from an arithmetic operation to take effect you need to reassign the variable.

a=a+5; #This will reassign value of a to the result of the operation

a+5; #This will not

The exceptions:

Chars: You can add two chars together and the result will be a string that concatenates both chars in order.

a=‘a’ + ‘b’; #a will evaluate to “ab”

a=”6” + 6; #a will evaluate to “66”

a=6+”6”; #a will evaluate to 12

a=6.0 + “6”; #a will evaluate to 12.0

a=”ab” + ‘c’; #a will evaluate to “abc”

12. Grouping

( )

Parentheses can be used to give precedence to expressions that are inside a parentheses.

5\*3-2; #This would evaluate to 13

5\*(3-2); #This would evaluate to 5

**Task 4**: Perform an arithmetic operation and print the result. Is the result what you expected?

13. While loop

While loops are made the same way as in Java. A Boolean value or something that evaluates to a Boolean value like a comparison is placed in parentheses after the while, and code in brackets is executed as long as the condition evaluates to true.

While ( condition ) {

While condition is true, code in this block will be executed forever

}

Incrementing and decrementing a number via a loop requires performing an addition or subtraction with itself like so

i=0; #this would be before the loop

i=i+1; #This would be inside the loop

14. Repeat

Repeat expressions can be used to repeat certain code. It is a truncated and extended do-while loop, so it runs the code once and repeats a set numver of times.

repeat(5) {

some code

}

#Code in bracket will be run 6 times. One initial time and 5 repetitions.

**Task 5**: Write a program involving either looping method what will print the numbers 1 to 10.

15. Blocks and scoping

Beyond controlling flow of a program, blocking is also used to control scope. EZlang follows Java for scoping so any code in a pair of brackets is local to only that scope with one exception. You cannot declare and initialize a new variable with the same name as a variable in an outside scope. Doing this will simply assign the outside variable the value assigned to the variable with the same name in the local scope like in Java.

x=5;print x;{ x=3;} print x; #This will print 5 and then print 3 since the local scope variable has the same name, the code in the bracket is reassigning value rather than declaring a new variable.

16. Manipulating arrays

Like in Java, subscripts are used to access elements in an array. The subscript for an array consists of the name of the array and square brackets, [ ], with the desired array index for the element you want to access. The first element in an array is index 0 just like Java.

array = [1,2,3,4,5];

print array[0]; #This will print 1

You can also use subscripts to change the element of an array of a particular index.

array = [1,2,3,4,5];

array[0]=0;

print array[0]; #This will print 0

For arrays the append operator is used to append elements to the end of an array. If you add two arrays together the result is an array where the right operand array is appended to the end of the left operand array.

a= [1,2,3];

a=a append [4,5,6]; # This evaluates to [1,2,3,4,5,6]

If you add any value and an array, the value is appended to the end of the array.

a=[1,2] append 3; #This evaluates to [1,2,3]

a=3 append [1,2]; #This also evaluates to [1,2,3]

The shrink operator is used to shrink an array by removing elements from the end of an array based on the right operand of a shrink operation. You cannot shrink an array beyond 0.

a=[1,2,3] shrink 1; #This evaluates to [1,2]

a=[1,2,3] shrink 3; #This evaluates to an empty array [].

a=[1,2,3] shrink 4; #this would result in a runtime error since you’re trying to shtink past 0

**Task 6:**

Create an array of your liking and assign it to a variable. Append the 0th element in the array to the array you made. Lastly, print the array. Is the result what you expected?

**Final Task:**

Given an array,

Array = [1,2,3,4,5,6,7,8,9,10];

Write a program that takes in user input for a variable and checks to see whether the entered input exists in the above array. If it does, print the string “It exists!”. If it doesn’t, append the user entered value to the array, and print the new array. You’ll need the length of the array which is 10.

**Bonus Final Task: FIZZBUZZ!**

Print the numbers 1 to 100. But for every number that is a multiple of 3 print “Fizz” instead of the number. For every number that is a multiple of 5 print “Buzz” instead of the number. For every number that is a multiple of 3 and 5, print “FizzBuzz” instead of the number.

Congratulations, you are one of the top 10 EZlang programmers in the world!