# **Capstone Portfolio**

Aleksandr Means (Team Member 2) & Wei You (Team Member 1)

CSCI 468 Spring 2023

Wei You

Section 1: SRC Zip file is attached

### Section 2:

Team member 2 contributed three tests and the documentation to our project, while I was responsible for the implementation. The documentation he provided me with is located in section 4 and the tests he provided me with are located below. The three tests are also located within the demo folder of the test folder of the project, with the class name of partnerTest, and the provided documentation can be found in the capstone folder named catscript.md. The estimated time contribution was about 50% 50% because it took about equal time and effort to get this project completed successfully and simply because of the ways that we used to complete it.

```
@Test
public void testingParenthesis() {
   AdditiveExpression expr = parseExpression( source: "1 + (3 + 5)");
   assertTrue(expr.isAdd());
   assertTrue(expr.getLeftHandSide() instanceof IntegerLiteralExpression);
   assertTrue(expr.getRightHandSide() instanceof ParenthesizedExpression);
void figureight() {
   assertEquals( expected: "10\n9\n8\n7\n6\n5\n4\n3\n2\n1\n0\n", compile(
            src: "function left(x : int) : int { if (x != 0) {print(x) return right(x - 1)} else {return 0} }n" +
                    "print(left(10))\n"));
@Test
public void parseListLiteralExpression() {
   ListLiteralExpression expr = parseExpression( source: "[[2, 3, 4], [2], [3, 68, 91, 81]]");
   assertEquals( expected: 3, expr.getValues().size());
   ListLiteralExpression inside = (ListLiteralExpression) expr.getValues().get(0);
   assertEquals( expected: 3, inside.getValues().size());
   ListLiteralExpression inside2 = (ListLiteralExpression) expr.getValues().get(1);
   assertEquals( expected: 1, inside2.getValues().size());
   ListLiteralExpression inside3 = (ListLiteralExpression) expr.getValues().get(2);
   assertEquals( expected: 4, inside3.getValues().size());
```

### Section 3:

The memoization pattern in the getListType method of the CatscriptType class in the compilers project utilizes a HashMap to cache instances of ListType. This approach ensures that when a ListType is requested for a specific CatscriptType, it is created only once and reused on subsequent requests, rather than being recreated each time. This method improves efficiency by reducing the number of object creations and the associated overhead, especially in performance-critical applications where getListType might be called frequently. Moreover, it helps in maintaining a smaller memory footprint since identical ListType objects are not needlessly duplicated. By using memoization, the system can scale better and operate faster as it avoids the computational cost of repeated instantiation.

```
static HashMap<CatscriptType, ListType> cache = new HashMap<>();
3 usages new *
public static ListType getListType(CatscriptType type) {
    return cache.computeIfAbsent(type, t -> new ListType(t));
}
```

## Section 4:

This is the documentation of the CatScript program that was provided by Team member

2.

2	31 ### AdditiveExpression
3 WW Introduction A Determint is a scription language that is easy to learn and binbly flexible to used The language summers the use of i	32 <u>Catscript</u> supports the use of the '+' operator for arithmetic addition and string concatenation, as well as the use of a 33 the '-' operator for arithmetic subtraction.
5 simple set of expression types, addition and multiplication operators, the ability to print to the standard	
output, as well as basic control flow abstractions. 7	35     Var X = 1 + 1     // X VVarVarGes CO Z       36     var y = 4 - 3     // y evaluates to 1
8 NR Features	<pre>37 var z = "hello " + "world" // z evaluates to "hello world" 38</pre>
10 ### Type System	39
Catscript supports a small simple set of expression types. The compiler uses type inference, but the type for the     variable can be declared.	40 ### Multiplication and Division 41 <u>Categript</u> also supports the use of the `*` operator for multiplication, and the `/` for division. Note that division
13	42 will result in an integer value, so any non-integer evaluation will truncate the remainder of the result.
14 * 'IntegerLiteralExpression' - 32-bit integer expressions, which can be modified by simple arithmetic operations. 15 * 'BooleanLiteralExpression' - simple true/false expressions.	44 var x = 3 * 3 // x evaluates to 9
16 * "StringLiteralExpression" - string expressions like those used in Java or C. 17 * "NulligenalExpression" - used to simify when constrting is unserious on annual	4S var y = 5 / 2 // y evaluates to 2 46 ····
18 * 'ListLiteralExpression' - a list of elements with a specified type, if given. Lists are immutable once declared.	47
19 20 ### IdentifierExpression	Comparison expressions can be made using '>', '<', '>=', and '<='. This allows a programmer to compare the values of
	50 integers in a script
	52 van x = 3
24 Variables in <u>Catscript</u> are assigned using the 'var' command. If explicit type declaration is desired, a programmer may 25 also format the statement like so: 'var foo : type'. This allows stricter definition of types when needed.	y os vary≡a 54
26	55 x >= x // true 56 x > y // false
27 var x : int = 32 28 var y : string = "string"	57 x <= y // true
	58 x < x // false 59 ····
31 ### AdditiveExpression	60 A1 ### Enunlit#Expression
32 <u>Catscript</u> supports the use of the '+' operator for arithmetic addition and string concatenation, as well as the use of 33 the '-' operator for arithmetic subtraction.	62 Equality expressions, using '==', allow a programmer to compare any two objects and determine if they are equal to one
	another. This includes null values, strings, integers, booleans, and lists. Furthermore, inequality expressions are als
61 ### EqualityExpression	89 89 MAN SyntaxErrorExpression
62 Equality expressions, using `==`, allow a programmer to compare any two objects and determine if they are equal to one 45 another. This includes cull values string integers hadrons and lists. Emphasized formula the 45 another the second string integers and the second string integers.	PO The <u>Categorint</u> compiler uses Syntax Error Expressions to help the programmer find errors in code during evaluation.
another. His includes nott values, strings, integers, booteans, and lists. Furthermore, inequality expressions are al 66 supported, using `!=`.	92 var x : list <int 2,="" 3]="" =="" [1,="" a="" be="" error<="" is="" notified="" p="" syntax="" that="" the="" there="" user="" will=""></int>
65 (***	93 A 4
	95
	96 ### PrintStatement 97 Print calls in <u>Catscript</u> allow the user to send text to the standard output. The print call formats the text to a new
	98 line.
	<pre>&gt;&gt;&gt; print("hello world")</pre>
(73 MMM UnaryExpression 74 Catscript has two unary operators. Using `not`, a user can express a logical negation of a boolean expression.	101 will print
75 Meanwhile, using `-` negates an integer expression.	103
	104 hetto world 105 ***
78 x = -x // x evaluates to -2	
79 var notPos : bool = not x > 0 // 'x > 0' evaluates to 'false', thus 'not x > 0' evaluates to 'true'	106 187 ### T#Statement
29 var notPos : Bool * not x > 0 // 'x > 0' evaluates to 'false', thus 'not x > 0' evaluates to 'true' 80	100 17 ### IfStatement 100 <u>Estacript</u> includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes if-else control flow statements, allowing a programmer to create conditional branching algorithms in 100 Estacript includes in the control flow statements in the control flow statement in the control flow statements in the control flow statement
79 var notPos : bool = not x > 0 // 'x > 0' evaluates to 'false', thus 'not x > 0' evaluates to 'true' 80 ''' 81 81 848 ParenthesizedExpression	10 des IfStatement Category includes if-else control flow statements, allowing a programmer to create conditional branching algorithms is 10 necessary. The statement requires a boolean expression mechaed within aparentheses immediately following the 'if and allows the programmer to define a boog of stratements that are enclosed within ourly branes. An optional 'lea' can
77 var notfös : bööl = not x > 0 // 'x > 0' evaluates to 'false', thos 'not x > 0' evaluates to 'true' 2020 2021 // 'x > 0' evaluates to 'false', thos 'not x > 0' evaluates to 'true' 2021 // 'x > 0' evaluates to 's evaluates to evaluate to 'false', thos 'not x > 0' evaluates to 'true' 2021 // 'x > 0' evaluates to 's evaluates to 'false', thos 'not x > 0' evaluates to 'true' 2021 // 'x > 0' evaluates to 's evaluates to 'false', thos 'not x > 0' evaluates to 'true' 2021 // 'x > 0' evaluates to 's evaluates to 'false', thos 'not x > 0' evaluates to 'false', thos 'not x > 0' evaluates to 'true' 2021 // 'x > 0' evaluates to 'false', thos 'not x > 0' evaluates to 'false', thos 'not x > 0' evaluates to 'true' 2021 // 'x > 0' evaluates to 'false', thos 'not x > 0' evaluates to 'false', the 'f	eff ification           intercipt incluses if-alse control flow statements, allowing a programmer to create conditional branching algorithms if         necessary. The statement requires a boltane expression enclosed within prestness immulately following the 'if' and         necessary. The statement requires a boltane expression enclosed within curly branch, an optimal else 'an         net of the analysis of a statement to define a body of statements that are enclosed within curly branch. An optimal else 'an         folks, and must also be folked by a body of statements contained within curly brance.
7/ var nothos : bool = not x > 0 // 'x > 0' evaluates to 'false', thus 'not x > 0' evlauates to 'true' 4 ## ForenthesizedExpression 2 Expression can las be written with parentheses to establish a desired precedence or to reduce ambiguity in more 4 complex expressions when mecessary. 5 ***	Aff IfStatement           IfStatement         Instancing algorithms is           Instancing         Instancing
<pre>// var notPos : bool * not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' // ***********************************</pre>	Interface         Interface           definition         relation in the statement of the statements, allowing a programmer to create conditional branching algorithms if measures. Interface is a statement requires a bolican expression enclosed within parentheses immediately following the 'If' and allows the programmer to define a body of statements that are enclosed within curly braces. An optional 'else' can follow and must also be followed by a body of statements contained within curly braces.
77 var nothos : bool = not x > 0 // 'x > 0 " evaluates to 'false', thus 'not x > 0 " evaluates to 'true' 18 18 18 18 19 19 10 10 10 10 10 10 10 10 10 10	### 1/Statewart     ### 1/Statewart     ### 1/Statewart     ### 1/Statewart     ### 1/Statewart     #################################
<pre>var notFos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true'  ### ParenthesizetCorression Expressions can also be written with parentheses to establish a desized precedence or to reduce ambiguity in more complex expressions when necessary. var x : bool = ('int '* e ('int '* 1)) == true // x evaluates to 'true'  var x : bool = ('int '* e ('int '* 1)) == true // x evaluates to 'true'  ### SyntactrueTorression The (trueTorrestion complex com</pre>	Interpretation           effective         fittering is aclues if-alse control flow statements, allowing a programmer to create conditional branching algorithms if measures. The statement requires a boltom expression enclosed within parentheses immediately following the 'if' and 'if' and the programmer to define a body of statements that are enclosed within curly braces. An optional else 'an enclosed within curly braces.           11            12            13         afford and most also be followed by a body of statements contained within curly braces.           13            14            15            16            17            18            19            10            11
<pre>var notFos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thos 'not x &gt; 0' evaluates to 'true' ### TorenthesizedEpression Expressions can also be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex symmetries with measure. wr x : bool = ('ist ] == ('ist '+ 1)) == true // x evaluates to 'true' ### SymtoxFromEpression The <u>External FromEprecedenceSion</u> The <u>External FromEprecedenceSion</u> Wr x : lations = [1, 2, 3] // the user will be notified that there is a south error</pre>	Image: control flow statements allowing a programmer to create conditional branching algorithms is included within pre-themes annotatively relation of the 'if' and included within curve branching algorithms is algorithm in 'if' and included within curve branching algorithms is an other 'if' and 'older and states at the programmer to define a body of statements that are enclosed within curve branching algorithms is 'or' and 'if' and 'older algorithm' if' algorithm' if' and 'older algorithm' if' algorithm' algorithm is 'or' of 'if' and 'older algorithm' if' algorithm' algorithm
<pre>var notFos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true'  ### ParenthesizedExpression Expressions can also be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex expressions when necessary.  ### SyntaxErverExpression The Cattering to complex cupiter uses Syntax Erver Expressions to help the programmer find ervers in code during evaluation.  ### SyntaxErverExpression The Cattering to complex cupiter uses Syntax Erver Expressions to help the programmer find ervers in code during evaluation.  ***********************************</pre>	### IfStatement         Catacrist incluses I-alse control flow statements, allowing a programmer to create conditional branching algorithms if necessary. The statement requires a boolean expression unclosed within parentheses ismediately following the 'if' and allows the programmer to define a body of statements that are enclosed within curly braces. An optional 'else' can followed by a body of statements contained within curly braces.   <
<pre>var nothos : bool = not x &gt; 0 // 'x &gt; 0" evaluates to 'false', thus 'not x &gt; 0" evaluates to 'true' var nothos : bool = not x &gt; 0 // 'x &gt; 0" evaluates to 'false', thus 'not x &gt; 0" evaluates to 'true' var x : bool = ('int i' == ('int '+ ii)) == true // x evaluates to 'true' var x : bool = ('int i' == ('int '+ ii)) == true // x evaluates to 'true' var x : bool = ('int i' == ('int '+ ii)) == true // x evaluates to 'true' var x : bool = ('int i' == ('int '+ ii)) == true // x evaluates to 'true' var x : bool = ('int i' == ('int '+ ii)) == true // x evaluates to 'true' var x : bool = ('int i' == ('int '+ ii)) == true // x evaluates to 'true' var x : bistic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there is a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there it a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there it a syntax error var x : listic it = [1, 2, 3] // the user will be notified that there it a syntax error var y : listic it = [1, 2, 3] // the user will be notified that there it a syntax error y = 0 // var x : listic it = [1, 2, 3] // the user will be not fiel there it = a syntax error y = 0 // var y =</pre>	### IfStatement         Catacrist inclues I-alse control flow statements, allowing a programmer to create conditional branching algorithms if necessary. The statement requires a boolean expression mnCosed within parentheses ismediately following the 'If' and allows the programmer to define a body of statements that are enclosed within curly braces. An optional "else" can follow and by a body of statements contained within curly braces.   <
<pre>var hotbos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true'  ### ParenthesizedEpression Expressions can also be written with parentheses to establish a desired precedence or to reduce subiguity in more complex expressions shan necessary.  var x : bool = ('int '* = ('int '* s)) == true // x evaluates to 'true'  var x : bool = ('int '* = ('int '* s)) == true // x evaluates to 'true'  ### SuftactroeExpression The (Extrept complet uses Syntax Error Expressions to help the programmer find errors in code during evaluation. var x : listint = [1, 2, 3] // the user will be notified that there is a syntax error  ### ForEntement For Binder and other control fing tool imideented in Categorian. They are in the for pack foreat one tage during the during to an advised of the time control imideented in Categorian. </pre>	First Induces         Induces           Gategingt incluses I-alse control flow statements, allowing a programmer to create conditional branching algorithms is necessary. The statement requires a bolean expression enclosed within parentheses immulately following the 1:1 and in allow the programmer to define a boly of statements that are enclosed within curly braces. An optional else can folke, and mast also be followed by a body of statements contained within curly braces.           In the intervention of the interventine of the intervention of the intervention of the interven
<pre>var hotfog : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thos 'not x &gt; 0' evaluates to 'true'  ### TorenthesizedCorresion Expressions can also be written with parentheses to establish a desized precedence or to reduce ambiguity in more complex expressions when necessary.  var x : bool = ('int i' == ('int '+ :)) == true // x evaluates to 'true'  var x : bool = ('int i' == ('int '+ :)) == true // x evaluates to 'true'  ### forfactorerExpression The (effecting complex uses Syntax Error Expressions to help the programmer find errors in code during evaluation.  var x : listicat = [1, 2, 3] // the user will be notified that there is a syntax error  A## forfactement For loops are another control. flow tool implemented in <u>Cationing</u>: They are in the for-each format, and are designed t it iterate our a list, storing each value to a uniquely defined imentified ent there is a the loop starting at the beginning of th iterate our a list, storing each value to a uniquely defined imentified ent the set is a syntax.</pre>	Application           Interpret incluses f-alse control flow statements, allowing a programmer to create conditional branching algorithms is necessary. The statement requires a boolean expression unclosed atthin parentheses lamediately following the 'lf' and labes the programmer to define a body of statements that are unclosed atthin curly braces. An optional relse can folke, and burst also be followed by a body of statements contained atthin curly braces.           Interpret in the statement of the stat
<pre>var notFos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thos 'not x &gt; 0' evaluates to 'true' ### TorenthesizedExpression Expressions can also be aritten alth parentheses to establish a desired precedence or to reduce ambiguity in more complex symmatisms and measury. user x : bool = ('int '=: ('int '+: 1)) ==: true // x evaluates to 'true' ### SymtoxTerreformession The <u>External complex</u> outpace to the symmatrix the more Expressions to help the programmer find errors in code during evaluation. '' '' '' '' '' '' '' '' '' '' '' '' ''</pre>	### ificitement         Cateringt inclumes if-alse control flow statements, allowing a programmer to create conditional branching algorithms if necessary. The statement requires a body of statements that are enclosed within ourly braces. An optional 'else' can follow abody of statements contained within ourly braces.
<pre>var hotVos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true'  ### ParenthesizedExpression Expressions can also be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex expression much measury.  **** **** **** **** **** **** ****</pre>	### IfStatement         Catacrost inclues I-alse control flow statements, allowing a programmer to create conditional branching algorithms if mecasary. The statement requires a boolean expression unclosed within parentheses ismediately following the 'if' and 'allow the programmer to define a body of statements that are enclosed within curly braces. An optional 'else' can follow allow a body of statements contained within curly braces. </td
<pre>var notFos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### ParenthesizedEpression Epressions can also be written with parentheses to establish a desired precedence or to reduce subjuity in more complax expressions when necessary var x : bool = ('int '* = ('int '* 1)) == true // x evaluates to 'true' war x : bool = ('int '* = ('int '* 1)) == true // x evaluates to 'true' ### Forticession Mediate Complax evaluates to reduce and provide the programmer find errors in code during evaluation war x : listcint = [1, 2, 3] // the user will be notified that there is a syntax error ### Forticessent For loops are another control flow tool implemented in Catigoright. They are in the for-each format, and are designed to List, and finally flowing each value to a uniquely defined identifier at each loop starting at the beginning of th List, and finally flowing each value to a uniquely defined identifier at each loop starting at the beginning of th List, and finally flowing each value to a uniquely defined identifier at each loop starting at the beginning of th List, and finally flowing each value to a uniquely defined identifier at each loop starting at the beginning of th List, and finally (inining when it has treated through each atkent for loops are subjected to a start of the list treated through each atkent</pre>	Production         Production           Participation         Second
<pre>var hotbog : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thos 'not x &gt; 0' evaluates to 'true' ### ParenthesizetCorression Expressions can also be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex expressions when necessary. ''' ''' ''' ''' ''' ''' '''' '''' ''</pre>	### Ifficience         If iterating isolates if also control flow statements, allowing a programmer to create conditional branching algorithms is necessary. The statement requires a boolean expression unclosed within curly braces. An optional relate an enclosed within curly braces.
<pre>var notFus : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true'  ### foreinthesizedSpression Expressions can take be written with parentheses to establish a desired precedence or to reduce ambiguity in more Complex expressions when necessary.  var x : bool = ('int if == ('int * + 1)) == true // x evaluates to 'true'  var x : bool = ('int if == ('int * + 1)) == true // x evaluates to 'true'  var x : bool = ('int if == ('int * + 1)) == true // x evaluates to 'true'  var x : bool = ('int if == ('int * + 1)) == true // x evaluates to 'true'  var x : bool = ('int if == ('int * + 1)) == true // x evaluates to 'true'  var x : bool = ('int if if == ('int * + 1)) == true // x evaluates to 'true'  var x : bool = ('int if if == ('int * + 1)) == true // x evaluates to 'true'  var x : listicity complier uses Syntax Error Expressions to help the programmer find errors in code during evaluation.  var x : listicit = [1, 2, 3] // the user will be notified that there is a syntax error  ### for/Gatement for loops are another control flow tool implemented in <u>Catsorigt</u>. They are in the for-each format, and are designed t ifterate one a list, storing each value to a unively defined identifier at each loop starting at the beginning of th list, and finally finishing when it has iterated through each elseent.  for (x in [1, 2, 3]) { print(x) }  will print  iterate</pre>	Image: A statement is a set of the statement, allowing a programmer to create conditional branching algorithms is necessary. The statement requires a booker expression enclosed within parentheses isomolitely following by a body of statements that are enclosed within curly braces. An optional relate can follow and must also be followed by a body of statements contained within curly braces.
<pre>ver inition : bool = not x &gt; 0 // 'x &gt; 0 evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true'  see forentlesizedEpression Epressions can also be aritten with parentheses to establish a desired precedence or to reduce ambiguity in more complax evareasions win measury.  var x: bool = ('int 1' == ('int ' + 1)) == true // x evaluates to 'true'  var x: bool = ('int 1' == ('int ' + 1)) == true // x evaluates to 'true'  var x: bool = ('int 1' == ('int ' + 1)) == true // x evaluates to 'true'  var x: bool = ('int 1' == ('int ' + 1)) == true // x evaluates to 'true'  var x: bool = ('int 1' == ('int ' + 1)) == true // x evaluates to 'true'  var x: listint = [1, 2, 3] // the user mill be notified that there is a syntax errer  *  ### forditement For loops are another control flow to a uniowity defined identifier at each loop starting at the beginning of th  list, and forally fraining men it has iterated through each elseent ### forditement for (in [1, 2, 3]) ( print(o) ) #### forditement if if an another control if an another if an</pre>	### IfStatement         Catacrist inclues I-alse control flow statements, allowing a programmer to create conditional branching algorithms if necessary. The statement requires a body of statements that are enclosed within ourly braces. An optional 'sis" can follow abody of statements that are enclosed within ourly braces.
<pre>var hoffos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### ParenthesizedEpression Epressions can also be written with parentheses to establish a desired precedence or to reduce ambiguity in more complax expressions when necessary</pre>	Image: A second seco
<pre>var hotbos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### Parenthesizenforcesion Expressions can take be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex expressions when necessary.  var x : bool = ('int ':= ('int ': 1)) == true // x evaluates to 'true'  ### SyntaxTrueTopression The (strengt complex uses Syntax Error Expressions to help the programmer find errors in code during evaluation.  var x : listint = [1, 2, 3] // the user will be notified that there is a syntax error  *** *** *** *** *** *** *** *** **</pre>	### ifficatement         idencipit Solutes if-alse control flow statements, allowing a programmer to create conditional branching algorithms if an excession. The statement requires a boolean expression enclosed within curly braces. An optional table of the abody of statements that are enclosed within curly braces. An optional table can enclosed within curly braces.  <
<pre>var hotbog : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'fatse', thos 'not x &gt; 0' evaluates to 'true' ### TorenthesizedCorression Expressions can take be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex expressions when necessary</pre>	### if/sizecont         Categrati Solumes i-alse control flow statements, allowing a programmer to create conditional branching algorithms if necessary. The statement requires a booken expression enclosed within parentheses lamediately folkaming the 'if' and 'and' and laws that programmer to define a book of statements that are enclosed within curly braces. An optional laws of the programmer to define a book of statements that are enclosed within curly braces.            if (5) {             print('true")             print('true")             print('not true")             print('not is true,' is true; the output mould be 'true'. However, if it is false, then the output is 'not true'.             print('not or) : moutput is a called as expressions in a script. A function call expression is used when a             revection fool : int ( return 0 )             print('not indisconterent             sitterent coll a subscher data is a script as well as make print statement calls, they can us             mectament coll a subscher data is no script as well as make print statement calls, they can us             satterent coll asust'scher data to be used as statements sinco th
<pre>ver intflus : bool = not x &gt; 0 // x &gt; 0 evaluates to 'false', thus 'not x &gt; 0' evaluates to 'frue' ### ParanthesizeExpression Expressions can tabe be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex evanesions winn escary. ''' ''''''''''''''''''''''''''''''''</pre>	### fifticecome         Catering: inclumes if-alse control flow statements, allowing a programmer to create conditional branching algorithms if necessary. The statement requires a book of statements that are enclosed within parentheses lamediately following high state in followed by a body of statements that are enclosed within ourly braces. An aptional relate is an approximate to define a body of statements that are enclosed within ourly braces.
<pre>ver hotFos : bool = not x &gt; 0 // x &gt; 0 evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### Parenthesize#Expression Expressions can labo be written with parentheses to establish a desired precedence or to reduce ambiguity in more complax evansions when measure. ''' ''' '''' '''''''''''''''''''''''</pre>	Image: A second seco
<pre>var hotbos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### ParenthesizefEquression Expressions can take be written with parentheses to establish a desired precedence or to reduce subjucty in more complex expressions when necessary. *** *** *** *** *** *** *** *** *** *</pre>	### #fitureson:         Categoint incluses if-alse control flow statements, allewing a programmer to create conditional branching algorithms if measures. The statement requires a bookern expression enclosed within curvy braces. An optional state can follow, and must also be followed by a body of statements that we enclosed within curvy braces. An optional state can follow and must also be followed by a body of statements that we enclosed within curvy braces.            if if is a more state of the a body of statements that we enclosed within curvy braces. An optional state can follow, and must also be followed by a body of statements contained within curvy braces.            if is a failed by a body of statements contained within curvy braces.            if is a failed by a body of statements contained within curvy braces.            if is a failed by a body of statements contained within curvy braces.            if is a failed by a body of statements contained within curvy braces.            if is a failed by a body of statements contained within curvy braces.            if is a failed by a body of statements and the statement failed by body of statements and the statement for the state of a statement for the state of a suble.            if the statement calls, they can be called to a suble.          if the statement calls, they can be a statement for the statement calls.            if the statement for the statement for the statement for the statement calls.          if the statement for the statement for the statemen
<pre>variantPost : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### ParenthesizetCorression Expressions can also be written with barentheses to establish a desired precedence or to reduce ambiguity in more camplex expressions when necessary. *** *** *** *** *** *** *** *** *** *</pre>	<pre>// ## fjSizeeourc files filesteen files the programme to define a body of statements, allesing a programmer to create conditional branching algorithms is necessary. The statement requires a body of statements that are enclosed stimin curly braces. An optional less the folds, and must also be followed by a body of statements that are enclosed stimin curly braces. // files the programmer to define a body of statements that are enclosed stimin curly braces. // files the programmer to define a body of statements that are enclosed stimin curly braces. // files the programmer to define a body of statements contained atthin curly braces. // files for the programmer to define a body of statements contained atthin curly braces. // files for the programmer to define a body of statements contained atthin curly braces. // files for the output if 's' is true; the output would be 'true'. However, if it is false, then the output is 'not true'. // files forction/files/</pre>
<pre>ver intfos : bool = not x &gt; 0 // x &gt; 0 evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ver intfos : bool = not x &gt; 0 // x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### ParanthesizeExpression Expression can tabe be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex evanesions winn accessary. *** *** *** *** *** *** *** *** *** *</pre>	<pre>/// /////////////////////////////////</pre>
<pre>ver hotFus : bool = not x &gt; 0 // x &gt; 0 evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### Parenthesize#Expression Expressions can labe be written with parentheses to establish a desired precedence or to reduce subjuity in more complex eventsions when messary. **** *******************************</pre>	<pre>/// // // // // // // // // // // // //</pre>
<pre>var hotbos : bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true'  /// ParanthesizeExpression Expressions can take be written with parentheses to establish a desired precedence or to reduce subjucty in more complax expressions when necessary. /// **********************************</pre>	<pre>/// // // // // // // // // // // // //</pre>
<pre>var hotPos: bool = not x &gt; 0 // 'x &gt; 0' evaluates to 'false', thus 'not x &gt; 0' evaluates to 'true' ### DerothesizedEncression Expressions can also be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex segression when necessary. *** *** *** *** *** *** *** *** *** *</pre>	<pre>## fificateon: discription: Second discription: The statement requires a booken expression enclosed within parentheses isendiately following the 'if' and allows the programme to define a body of statements that are enclosed within curly braces. An optional rate is en folds, and burstato is followed by a body of statements that are enclosed within curly braces. folds, and burstato is followed by a body of statements that are enclosed within curly braces. folds, and burstato is followed by a body of statements contained within curly braces. folds, and burstato is followed by a body of statements contained within curly braces. folds, and burstato is followed by a body of statements contained within curly braces. folds of the programme to define a body of statements contained within curly braces. folds the output if 's' is true; the output would be 'true'. However, if it is false, then the output is 'not true'. burst for the output if 's' is true; the output would be 'true'. However, if it is false, then the output is 'not true'. forection fold is not true of the output would be 'true'. However, if it is false, then the output is 'not true'. forection fold (); int ( furture 0 ) print('1 + ford ) : forection fold (); int ( furture 0 ) print('1 + ford ) : function fold (); int ( furture 0 ) forection for() ( furture 1); forection for() ( furture 1); forection for() ( furture 1); forection for() ( furture 1); forection for() ( furture 1); for (); for (); f</pre>
<pre>ver notifies : bool + not x &gt; 0 // x &gt; 0 vertuates to 'fates', thus 'not x &gt; 0 ' evaluates to 'true' ### ParanthesizedExpression Expressions can tabe be written with parentheses to establish a desired precedence or to reduce ambiguity in more complex everysions with measury. ''' ''' ''' ''' '''' '''''''''''''''</pre>	<i>eff fifthetement</i> Glassing isclues if clace control flow statements, allowing a programmer to create conditional branching algorithms if             recessary. The statement requires a booker expression enclosed atthin parentheses ismediately following the 'if' and             low durat also be followed by a body of statements that are enclosed atthin curly braces. An optional relate can             fold, and must also be followed by a body of statements contained atthin curly braces.                 if (0) (

### Section 5:

This is the UML diagram for the recursive descent parsing algorithm utilized by the compiler's parsing project.







### Section 6:

When comparing parser generators and recursive descent parsers, the choice often depends on the specific needs of the language being implemented and the preferences of the developers. Parser generators, like ANTLR or Yacc, provide a powerful tool for creating parsers from a high-level description of the grammar. These tools automatically generate the parsing code based on grammar rules defined in a formal syntax, often BNF (Backus-Naur Form). This can accelerate development and ensure that the grammar is correctly and consistently implemented, as changes to the grammar are centrally managed and propagated through the generated code. For a language like Catscript, which seems to have a structured type system and potentially complex syntax rules, a parser generator can offer a robust framework to handle parsing efficiently, minimizing manual coding errors and inconsistencies.

On the other hand, recursive descent parsers are hand-coded and provide the developer with finer control over the parsing process. This method involves writing functions or methods that directly implement the grammar rules. Each function typically corresponds to a non-terminal in the grammar, and these functions call each other recursively to match the input tokens against the grammar rules. Recursive descent parsers are particularly advantageous when the language features context-sensitive constructs or when the parsing logic needs to intertwine tightly with other aspects of the interpreter or compiler, such as type checking and error handling. In the context of Catscript, if the language demands intricate error reporting or specific handling of its unique types like CatscriptType and ListType, a recursive descent parser provides the necessary control to integrate these features seamlessly. Additionally, since recursive descent parsers do not require external tools, they are preferred for smaller or more experimental languages where external dependencies are to be minimized.

#### Section 7:

Test Driven Development (TDD) has been a central strategy in our approach to developing the capstone project, shaping how we plan, write, and verify our code. Fundamentally, TDD involves writing tests for specific functionalities before even writing the code that implements those functionalities. This method promotes a well-organized development process where requirements are translated into specific test cases, ensuring that all features are thoroughly planned and tested from the outset. Personally, I have found TDD immensely beneficial as it compels the team to clarify requirements and establish clear, measurable objectives for every feature before proceeding with implementation. This upfront investment in testing helps avoid the common pitfall of feature creep and ensures high coverage and code quality from early in the project lifecycle.

From a personal perspective, TDD aligns well with my methodical approach to problem-solving. It allows me to focus intensely on one specific aspect of the system at a time, ensuring that each piece of functionality is correctly implemented and robust against future changes before moving on. This approach to building a project not only enhances my sense of accomplishment but also instills a discipline of continuous verification, which is crucial for maintaining long-term code health. Moreover, TDD's emphasis on testing first reduces the likelihood of encountering severe bugs later in the development cycle, which can be both demoralizing and costly to fix. While TDD can sometimes slow the initial stages of development as tests are formulated and refined, I find that this is mitigated by the smoother and quicker integration phases, as well-developed tests pave the way for integrating complex systems more reliably.