https://www.google.com/search?q=manatee&biw=1242&bih=522&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjn987L1vTRAhVCySYKHUHiAcMQ_AUIBigB#imgrc=lZU3VsrZH0F2WM:

ManiT

Manager: Akiva Dollin - acd2164
Language Guru: Irwin Li - izl2000
System Architect: Seungmin Lee - sl3254
Tester: Dong Hyeon (Paul) Seo - ds3457

INTRODUCTION:

ManiT is a C-like language which enables easy manipulation of large integers and linear systems and compiles into LLVM. This language would be extremely useful specifically for quick manipulation of 2D and 3D graphs as well as general purpose number manipulation.

DESCRIPTION:

This language is useful because it allows the programmer to manipulate data types with ease. Because of this functionality, ManiT is great for implementing system based graphing programs. Imagine a 3D figure that is represented by a system of equations. With ManiT a specific coefficient within this system can be manipulated quickly and easily without the need to rewrite the equations.

Additionally, this could be useful for matrix manipulation. Consider robotic path planning. A robot's global and relative position is represented by a 3x3 matrix. By manipulating a specific value within that matrix, the robot's position and orientation changes. ManiT makes this process simple and straightforward.

ManiT will be a statically typed language that uses C-like syntax and Python-like indexing for arrays, ints, and strings. Variable scope will also be C-like (enclosed by {}).

BUILT-IN TYPES:

Type	**Description**	**Create**
Integer	Numbers without a decimal point (also known as Integers)	int a = 4 int b =13
Float	Number that contains a decimal point.	float c = 4.23 float d = 5.7888
String	A sequence of characters	String e = “StephenEdwardRocks”
Array	A container to hold multiple data of the same datatype	int[] f = [1, 2, 3]
Character	A character variable	char g = ‘4’ char h = ‘h’
Equation	Defines a string of variables,coefficients, and exponents.	Eqn temp = “10x^2+4y+3” Eqn name = “equation string”

FEATURES:

Features	**Type supported --- what it does**
.left / .right	float ---- Returns the number left/right of the decimal point
.pow	int / float / String ---- Returns the exponent
.coeff	Int / float ---- Returns the coefficient
a[start:end:increment]	int / float / String / array ---- Extracts certain sequence of characters from a variable
a * n	int / float / String / array ---- Returns the amount of a n times e.g. “hi” * 3 == “hihihi” 3 * 3 == 9
$a[x_1][x_2][x_3]...[x_n]$ $b[n]$ $([x_1][x_2][x_3]...[x_n])$	int / float / String / array ---- Returns the indexes, x_1, x_2, x_3, x_n If index is negative, it returns the indexes right to left.

	e.g. “hello”[-1] == ‘o’
graph(String title, Eqn equation)	Graphs an equation with a title using OpenGL.

OPERATORS:

Symbol	**Action**
+, - , * , / , **	Math operations: Add, Subtract, Multiply, Divide, Exponent.
=, +=, -=	Assignment Operations: Assign, Increment, Decrement.
==, !=, >, <, >=, <=	Boolean Operations: Equal to, Not equal to, Greater than, Less than, Greater than or equal to, Less than or equal to.
//, /* block comment */	Comments: Single-line, Block.
&&, ||, !	Logical Operators: AND operator, OR operator, NOT operator.
++, --	Increment Operators: Increment by 1, Decrement by 1.
%	Remainder Operator

CONTROL FLOW:

Type	**Structure**
If / else if / else	if () {} elseif () {} else {}
For loop	for () {}
While	while () {}

FUNCTION DECLARATION:

```
//template
return_type function_name(param_type param_name, ...) {
      // body of function
}
//example
int add(int a, int b) {
      return (a + b);
}
```

Some functions perform the desired operations without a return type. In this case, the `return_type` keyword will be **`void`**.

Some functions perform the desired operations without parameters. In this case, simply put the keyword **`void`** inside the parenthesis instead of a `param_type` and a `param_name`.

EXAMPLE PROGRAM 1:

```
//example program
//define equations
Eqn temp = "10x^2+4y+3";
Eqn temp2 = "6x^2+4y+3";

//graph the equations
graph("Graph 1", temp);
graph("Graph 2", temp2);

//iterate and manipulate graphs
for( int i=0; i<10; i++){
      /* Go to index 2 of temp, 3, get the constant side of '3',
            and change constant to i */
      temp[2].coeff = i;

      /* Go to index 1 of temp, 4y, get the variable side of 'y',
            and change power to 3
            4y → 4y^3 */
      temp[1].pow[y] = 3;

      /* Go to index 0 of temp, 10x^2, get the variable side of 'z',
            Since not exist create z and its power is 1
            10x^2 → 10x^2z^1 */
      temp[0].pow[z] == 1; //
```

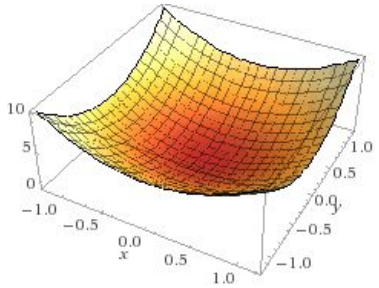
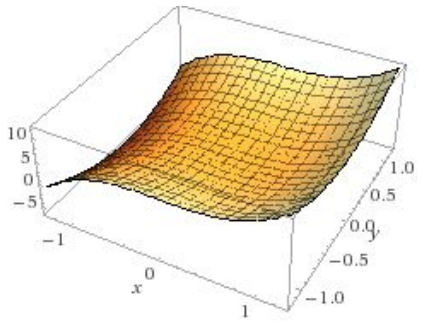
```
      //temp[0] == 10x^2
      //temp[0].coeff.exp[x] = 4;  <-->  equals 10x^4
      //temp[1] == 4x
      //temp[1].coeff == 4

      /* Displays graph */
      graph("Graph "+i, temp);
}

//Eqn temp = ["14.321x^2+4y+3"];
//temp[0].coeff.right[1] = 3; //now 14.331
```

EXAMPLE PROGRAM 2:

CODE	**Result**

//another example program with 3D //equations
Eqn temp = ["3x^2+4y^2"];
graph("Graph 1", temp);
 	
temp[0].exp[x] = 3;
graph("Graph 2", temp);
	Graph 1:
	 
	
	Graph 2:
	 

EXAMPLE PROGRAM 3:

```
// Example program which converts to binary using ManiT indexing
int convertToBinary(int num)
{
      int binary = 0;
      int i = 0;
      while(num != 0) {
            remainder = num % 2;
            /* integer division */
            num = num / 2;
            /* change first digit to 2 digit number for more space */
            binary[0] = 10 + remainder;
      }
      binary[0] = 0;
      return binary;
}
```

EXAMPLE PROGRAM 4:

```
// Example program in which the user is able to utilize the various
// flexibility of data types such as int to simplify
// computations of otherwise very large calculations
// that cannot be stored in int properly in other traditional
// languages

// Using Fermat's Little Theorem
// Computing:(5^4260603732 - 3^4260603732) mod 19
int a = 5^4260603732
int b = 3^4260603732
a.pow == 4260603732 // a's power is equal to 4260603732
b.pow == 4260603732 // b's power is equal to 4260603732
int n = 19

// a^(p-1) and 1 are congruence modulo p for p prime
// b^(p-1) and 1 are congruence modulo p for p prime

// Then
// 5^18 and 1 are congruence modulo 19
// 3^18 and 1 are congruence modulo 19

// a.pow = 6
// b.pow = 6
a.pow = a.pow % (n-1)
```

```
b.pow = b.pow % (n-1)

// (5^4260603732 - 3^4260603732) mod 19
// =
// (5^6 - 3^6) mod 19

// c = (5^6 - 3^6)
int c = a - b
int result = c % 19
print(result)
```