**Assignment No -**

**Problem statement**

Program in C++ to create a class LOG having two data members x and y to represent x logb y where the base b is a constant given by the user.Perform four basic operations: addition, subtraction, power, root using operator overloading.

**Algorithm**

* **Name of the class :-** Log
* **Private data members of the class :-**x, y
* **Public member functions of the class :**
  + operator+()
  + operator-()
  + power()
  + Log():x(0),y(0) //Constructor

**Algorithm for method operator+():**

1. Set p1 = pow(y,x)
2. Set p2 = pow( b.y , b.x )
3. Call Log n( 1, p1\*p2 )
4. Return n

**Algorithm for method operator-():**

1. Set p1 = pow(y,x)
2. Set p2 = pow( b.y , b.x )
3. Call Log n( 1, p1/p2 )
4. Return n

**Algorithm for method power():**

1. Set p1 = pow ( y,x )
2. Call Log n( 1,p1 )
3. Return n

**Algorithm for method main():-**

1. Print "Enter x and y for the first term in the form (x log y)";
2. Read x,y
3. Set l1 = Log( x,y )
4. Print “Enter x and y for the second term in the form (x log y)";
5. Read x,y
6. Set l2 = Log( x,y )
7. Print “l1 + l2”
8. Set Log l3 = l1 + l2
9. Print “l3”
10. Set l3 = l1 – l2
11. Print “l1 – l2”
12. Print “Applying power rule :”
13. Set l3 = l1.power()
14. Print “l1 -> l3”
15. Set l3 = l2.power()
16. Print “l2 -> l3”

**Source Code**

#include <iostream>

#include <math.h>

using namespace std;

class Log {

private:

double x, y; // x log y

public:

Log() : x(0), y(0) {}

Log(double \_x, double \_y) : x(\_x), y(\_y) {}

Log operator+(Log &b) {

double p1 = pow(y, x);

double p2 = pow(b.y, b.x);

Log n(1, p1\*p2);

return n;

}

Log operator-(Log &b) {

double p1 = pow(y, x);

double p2 = pow(b.y, b.x);

Log n(1, p1/p2);

return n;

}

Log power() {

double p1 = pow(y, x);

Log n(1, p1);

return n;

}

friendostream& operator<<(ostream&os, Log &l);

};

ostream& operator<<(ostream&os, Log &l) {

returnos<<l.x<< " log " <<l.y;

}

int main() {

double x, y;

Log l1, l2;

cout<< "Enter x and y for the first term in the form (x log y)" <<endl;

cout<< "Enter x : ";

cin>> x;

cout<< "Enter y : ";

cin>> y;

l1 = Log(x, y);

cout<< "Enter x and y for the second term in the form (x log y)" <<endl;

cout<< "Enter x : ";

cin>> x;

cout<< "Enter y : ";

cin>> y;

l2 = Log(x, y);

cout<< l1 << " + " << l2 << " : ";

Log l3 = l1 + l2;

cout<< l3 <<endl;

l3 = l1 - l2;

cout<< l1 << " - " << l2 << " : " << l3 <<endl;

cout<< "Applying power rule : " <<endl;

l3 = l1.power();

cout<< l1 << " -> " << l3 <<endl;

l3 = l2.power();

cout<< l2 << " -> " << l3 <<endl;

}

**Input & Output**

**Set 1**

Enter x and y for the first term in the form (x log y)

Enter x : 2

Enter y : 5

Enter x and y for the second term in the form (x log y)

Enter x : 5

Enter y : 8

2 log 5 + 5 log 8 : 1 log 819200

2 log 5 - 5 log 8 : 1 log 0.000762939

Applying power rule :

2 log 5 -> 1 log 25

5 log 8 -> 1 log 32768

**Set 2**

Enter x and y for the first term in the form (x log y)

Enter x : 1

Enter y : 1

Enter x and y for the second term in the form (x log y)

Enter x : 2

Enter y : 3

1 log 1 + 2 log 3 : 1 log 9

1 log 1 - 2 log 3 : 1 log 0.111111

Applying power rule :

1 log 1 -> 1 log 1

2 log 3 -> 1 log 9

**Discussion**

* Name of an operator function is always operator keyword followed by symbol of operator and operator functions are called when the corresponding operator is used.
* For operator overloading to work, at least one of the operands must be a user defined class object.
* We can also write conversion operators that can be used to convert one type to another type.