Calculator.h

```
#ifndef CALCULATOR_H
#define CALCULATOR_H


//represents template class for a calculator
template<class T>
class Calculator {
    private:
    T value1; //represents first value
    T value2; //represents second value
    public:
    Calculator() {value1 = T(); value2 = T();} //constructior: initializes values to default
    Calculator(T value1, T value2); //constructor: set values
    T getValue1(); //returns value1
    T getValue2(); //returns value2
    T getSum(); //uses arithmetic operatior '+'
    int getLogicalAND(); //uses logical operatior '&&'
    bool isGreater(); //uses relational operatior '>'
    
};

#endif
```

Calculator.cpp

```cpp
#include "Calculator.h"

//constructor that sets values
template<class T>
Calculator<T>::Calculator(T value1, T value2) {
    this->value1 = value1; //sets value1
    this->value2 = value2; //sets value2
}

template<class T>
T Calculator<T>::getValue1() {
    return value1; //returns value1
}

template<class T>
T Calculator<T>::getValue2() {
    return value2; //returns value2
}

template<class T>
T Calculator<T>::getSum() {
    return value1 + value2; //returns sum of value1 and value2
}

template<class T>
int Calculator<T>::getLogicalAND() {
    return value1 && value2; //reurns && of value1 and value2
}

template<class T>
bool Calculator<T>::isGreater() {
    return value1 > value2; //returns > of value1 and value2
}
```

CalcMain.cpp

```cpp
#include <iostream>
#include <string>
#include "Calculator.cpp"
using namespace std;

//represents arithmetic class
class Arithmetic {
    private:
    int intData; //for ints
    float floatData; //for floats
    double doubleData; //for doubles
    template<typename T2>
    /*
    a template function local to this class. The function can be
    called with different types of Calculator objects, and it prints out the values and calls
    getSum(), getLogicalAND(), and isGreater() functions from the template class
    to test the 3 different types of operations, and print the results from those operations as seen
    in the sample output below
    */
    void printOperations(T2 obj);

    public:
    Arithmetic() {intData = 0; floatData = 0; doubleData = 0;} //constructor: initializes all variables to 0
    Arithmetic(int i, float f, double d); //constructor: sets variables
    /*
    - creates a Calculator object of type int from the
    Calculator template class while setting the Calculator values to the local intData and the
    intData from the received object obj. It then calls printOperations(T2 obj)
    function with the newly created Calculator object to test the 3 different types of operations.
    */
    void intOperations(Arithmetic obj);
    /*
    creates a Calculator object of type float from the
    Calculator template class while setting the Calculator values to the local floatData and
    the floatData from the received object obj. It then calls printOperations(T2 obj)
    function with the newly created Calculator object to test the 3 different types of operations.
    */
    void floatOperations(Arithmetic obj);
```

```cpp
    /*
    creates a Calculator object of type double from
    the Calculator template class while setting the Calculator values to the local doubleData
    and the doubleData from the received object obj. It then calls printOperations(T2
    obj) function with the newly created Calculator object to test the 3 different types of
    operations.
    */
    void doubleOperations(Arithmetic obj);
    int returnInt(); //helper to return int
    float returnFloat(); //helper to return float
    double returnDouble(); //helper to return double
};

//sets variables of intData, floatData, and doubleData
Arithmetic::Arithmetic(int i, float f, double d) {
    this->intData = i;
    this->floatData = f;
    this->doubleData = d;
}

template<typename T2>
void Arithmetic::printOperations(T2 obj) {
    //line for sum
    cout << obj.getValue1() << " + " << obj.getValue2() << " = " << obj.getSum() << endl;
    //line for &&
    cout << obj.getValue1() << " && " << obj.getValue2() << " = " << obj.getLogicalAND() << endl;
    //because in sample output they wanted true and false instead of 1 and 0 which c++ automatically converts bools to
    string boolean;
    if(obj.isGreater() == 1)
        boolean = "true";
    else
        boolean = "false";
    //line for >
    cout << obj.getValue1() << " > " << obj.getValue2() << " = " << boolean << endl;
}

//returns intData
int Arithmetic::returnInt() {
    return intData;
}
```

```
//returns floatData
float Arithmetic::returnFloat() {
    return floatData;
}

//returns doubleData
double Arithmetic::returnDouble() {
    return doubleData;
}

void Arithmetic::intOperations(Arithmetic obj) {
    int temp = obj.returnInt();
    Calculator<int> calculator(intData, temp); //creates calculator object with both int values
    printOperations(calculator); //performs the operations with the ints
}

void Arithmetic::floatOperations(Arithmetic obj) {
    float temp = obj.returnFloat();
    Calculator<float> calculator(floatData, temp); //creates calculator object with both float values
    printOperations(calculator); //performs the operations with the floats
}

void Arithmetic::doubleOperations(Arithmetic obj) {
    double temp = obj.returnDouble();
    Calculator<double> calculator(doubleData, temp); //creates calculator object with both double value
    printOperations(calculator); //performs the operation with the doubles
}

int main() {
    // Create 1st object
    int int1 = 4;
    float f1 = 10.4;
    double d1 = 20.1;
    Arithmetic object1(int1, f1, d1);
```

```
    // Create 2nd object
    int int2 = 7;
    float f2 = 0.0;
    double d2 = 3.5;
    Arithmetic object2(int2, f2, d2);

    // Check operation on integer data type
    cout << "\nPrinting INTEGER operations..." << endl;
    object1.intOperations(object2);

    // Check operation on float data type
    cout << "\nPrinting FLOAT operations..." << endl;
    object1.floatOperations(object2);

    // Check operation on double data type
    cout << "\nPrinting DOUBLE operations..." << endl;
    object1.doubleOperations(object2);
} // end main
```

Makefile

```
# Calculator program Makefile

Calculator:  Calculator.o CalcMain.o
    g++ -o Calculator Calculator.o CalcMain.o

Calculator.o: Calculator.h Calculator.cpp
    g++ -c Calculator.cpp

CalcMain.o: Calculator.h CalcMain.cpp
    g++ -c CalcMain.cpp

clean:
    rm Calculator.o CalcMain.o Calculator
```

Console Output

```
-bash-4.2$ ./Calculator

Printing INTEGER operations...
4 + 7 = 11
4 && 7 = 1
4 > 7 = false

Printing FLOAT operations...
10.4 + 0 = 10.4
10.4 && 0 = 0
10.4 > 0 = true

Printing DOUBLE operations...
20.1 + 3.5 = 23.6
20.1 && 3.5 = 1
20.1 > 3.5 = true
-bash-4.2$
```