Temperature Class

1. Temperature.h

```
//////////////////////////////////////////
// Temperature.h
#ifndef __TEMPERATURE__
#define __TEMPERATURE__
#include <iostream>
using namespace std;

class Temperature {
private:
    double degree;
    char scale;
public:
    /*************************************************************
     * constructors
     */
    Temperature();
    Temperature(double d);
    Temperature(double d, char s);


    /*************************************************************
     * accessor functions
     */
    void setDegree(double d);
    void setScale(char s);
    double getDegree();
    char getScale();

    /*************************************************************
     * assignments
     */
    Temperature& operator=(const Temperature& right);
    //void operator=(int right);        // this can only do t1 = 3;
    Temperature& operator=(int right);  // this can do t2 = t1 = 3;
    void operator=(double right);
    void operator=(char right);

    /*************************************************************
     * arithmetic operators
     */
     /*************************************************************
      * operator+
      * t1+t2
      */
    Temperature operator+(const Temperature& right) const;

    /*************************************************************
     * operator+
     * t1+double
     */
    Temperature operator+(const double& right) const;

    /*************************************************************
```

```
     * operator+
     * double+t1
     */
    friend Temperature operator+(const double& left, const Temperature& right);

    /*************************************************************
     * unary prefix operator++
     * ++t
     */
    Temperature operator++();

    /*************************************************************
     * unary postfix operator++
     * t++
     */
    Temperature operator++(int);

    /*************************************************************
     * logical operators
     */
    bool operator==(const Temperature& right) const;
    bool operator<(const Temperature& right) const;
    bool operator==(const double& right) const;
    friend bool operator==(const double& left, const Temperature& right);

    /*************************************************************
     * friend operator<<
     * out << t
     */
    friend ostream& operator<<(ostream& out, const Temperature& t);

    /*************************************************************
     * friend operator>>
     * in >> t
     */
    friend istream& operator>>(istream& in, Temperature& t);

    /*************************************************************
     * conversion methods
     */
    void toC();
    void toF();
    void toK();

};

#endif
```

2. Temperature.cpp

```
////////////////////////////////////////
// Temperature.cpp
#include "Temperature.h"
#include <iostream>
using namespace std;

/*************************************************************
```

```
 * constructors
 */
Temperature::Temperature() {
    degree = 0;
    scale = 'C';
}

Temperature::Temperature(double d) {
    degree = d;
    scale = 'C';
}

Temperature::Temperature(double d, char s) {
    s = toupper(s);
    if (s == 'K' || s == 'C' || s == 'F') {
        degree = d;
        scale = s;
    } else {
        degree = 0;
        scale = 'C';
    }
}

/*************************************************************
 * accessor functions
 */
void Temperature::setDegree(double d) {
    degree = d;
}

void Temperature::setScale(char s) {
    if (s == 'K' || s == 'C' || s == 'F') {
        scale = s;
    }
}

double Temperature::getDegree() {
    return degree;
}

char Temperature::getScale() {
    return scale;
}

/*************************************************************
 * assignments
 */
Temperature& Temperature::operator=(const Temperature& right) {
    degree = right.degree;
    scale = right.scale;
    return *this;
}

Temperature& Temperature::operator=(int right) {
    degree = right;
    return *this;
}
```



```
void Temperature::operator=(double right) {
   degree = right;
}

void Temperature::operator=(char right) {
   scale = toupper(right);
}

/*************************************************************
 * arithmetic operators
 */
 /*************************************************************
  * operator+
  * t1+t2;
  */
Temperature Temperature::operator+(const Temperature& right) const {
    Temperature t = right;
    // convert right.scale to myself's scale
    switch (scale) {  // myself's scale
    case 'C':
        t.toC();
        break;
    case 'F':
        t.toF();
        break;
    case 'K':
        t.toK();
        break;
    }
    t.degree = degree + t.degree;
    return t;
}

/*************************************************************
 * operator+
 * t1+double
 */
Temperature Temperature::operator+(const double& right) const {
    return Temperature(degree + right, scale);
}

/*************************************************************
 * friend operator+
 * double+t1
 */
Temperature operator+(const double& left, const Temperature& right) {
    return Temperature(left + right.degree, right.scale);
}

/*************************************************************
 * unary prefix operator++
 * ++t;
 */
Temperature Temperature::operator++() {
    degree++;     // increment this object
    return *this; // return the incremented object
}
```



```
/*************************************************************
 * unary postfix operator++
 * t++;
 */
Temperature Temperature::operator++(int) {
    Temperature t(degree, scale); // save the orignal object
    degree++;                     // increment this object
    return t;                     // return the original object
}

/*************************************************************
 * logical operators
 */
bool Temperature::operator==(const Temperature& right) const {
    Temperature t1 = *this;
    Temperature t2 = right;
    t1.toC();
    t2.toC();
    if (t1.degree == t2.degree) {
        return true;
    }
    else {
        return false;
    }
}

bool Temperature::operator<(const Temperature& right) const {
    Temperature t1 = *this;
    Temperature t2 = right;
    t1.toC();
    t2.toC();
    if (t1.degree < t2.degree) {
        return true;
    }
    else {
        return false;
    }
}

bool Temperature::operator==(const double& right) const {
    if (degree == right) {
        return true;
    }
    else {
        return false;
    }
}

/*************************************************************
 * friend operator==
 * double == t;
 */
bool operator==(const double& left, const Temperature& right) {
    if (left == right.degree) {
        return true;
    }
    else {
        return false;
```

```
    }
}

/*************************************************************
 * friend operator<<
 * out << t;
 */
ostream& operator<<(ostream& out, const Temperature& t) {
    out << t.degree << "°" << t.scale;
    return out;
}

/*************************************************************
 * friend operator>>
 * cin >> t;
 */
istream& operator>>(istream& in, Temperature& t) {
    in >> t.degree >> t.scale;
    return in;
}

/*************************************************************
 * conversion methods
 */
void Temperature::toC() {
    switch (scale) {
    case 'F':
        degree = (degree - 32) * 5 / 9;
        break;
    case 'K':
        degree = degree - 273.15;
        break;
    }
    scale = 'C';
}

void Temperature::toF() {
    switch (scale) {
    case 'C':
        degree = degree * 9 / 5 + 32;
        break;
    case 'K':
        degree = (degree - 273.15) * 9 / 5 + 32;
        break;
    }
    scale = 'F';
}

void Temperature::toK() {
    switch (scale) {
    case 'C':
        degree = degree + 273.15;
        break;
    case 'F':
        degree = (degree - 32) * 5 / 9 + 273.15;
        break;
    }
    scale = 'K';
```

```
}
```

3. Main

```
#include <iostream>
using namespace std;
#include "Temperature.h"

int main() {
   Temperature t1(72, 'F'), t2, t3;

   t2.setDegree(12);
   t2.setScale('K');

   t3 = t2 = t1;
   t2.toK();
   cout << t1 << endl;
   cout << t2 << endl;
   // cout << ++t2 << endl;
   // cout << t3++ << endl;
   t3 = t1 + t2;
   cout << t3 << endl;
   t3 = t3 + 1;
   cout << t3 << endl;
   t3 = 1.2 + t3;
   cout << t3 << endl;

   }
```

Sample output.

```
72°F
295.372°K
144°F
145°F
146.2°F
```

4. Exercises (Problems with an asterisk are more difficult)

1. What are some other methods that you can add to the above Temperature class to make it more complete? Write the code for these new methods.