Name : M.Ahmed Arshed

Roll no : p17-6099

SECTION – A

Lab\_ Tasks 08(Operator\_Overloading)

Tasks # 01

#include<iostream>

#include<string>

#include<cmath>

using namespace std;

class Distance{

private:

int length ,inches;

public:

Distance()

{

length = 0;

inches = 0;

// cout<<"default constructor is called: " << endl;

}

Distance(int a ,int b)

{

a = length;

b = inches;

// cout<<"parameterized constructor is called:" <<endl;

}

void set\_distance(int a ,int b)

{

a = length;

b = inches;

}

void show\_distance(int a, int b)

{

cout << a << endl;

cout << b<<endl;

}

bool operator <(const Distance &d2)

{

if(length < d2.length && inches < d2.inches)

return true;

else

return false;

}

bool operator += (const Distance &d2)

{

length = length + d2.length;

inches = inches + d2.inches;

cout<<length << inches;

}

Distance operator ++(int)

{

//post form

d1.length = lenght ++;

d1.inches = inches ++;

return d1;

}

bool operator ++ (const Distance &d2)

{

d2.length = ++length;

d2.inches = ++inches;

return d2;

}

// += , -= , \*= , /= , %=;

bool operator -= (const Distance &d2)

{

length = length - d2.length;

inches = inches - d2.inches;

cout<<length << inches;

}

bool operator \*= (const Distance &d2)

{

length = length \* d2.length;

inches = inches \* d2.inches;

cout<<length << inches;

}

bool operator /= (const Distance &d2)

{

length = length / d2.length;

inches = inches / d2.inches;

cout<<length << inches;

}

bool operator %= (const Distance &d2)

{

length = length % d2.length;

inches = inches % d2.inches;

cout<<length << inches;

}

};

class time{

private:

int hr,min ,sec;

public:

void set\_time(int a ,int b,int c)

{

a = hr;

b = min;

c = sec;

}

bool operator <<(const time &t2)

{

cout<<t2;

}

bool operator >>(const time & t2,comp)

{

cin>>t2;

}

};

int main()

{

Distance d1 ,d2;

d1.set\_distance(4,5);

d1.show\_distance(4,5);

d1.operator <(d2);

d1.operator +=(d2);

d2.operator ++(d1);

d1.operator ++(d2);

d1.operator -=(d2);

d1.operator \*=(d2);

d1.operator /=(d2);

d1.operator %=(d2);

time t1 ,t2;

t1.set\_time(1,12,32);

t1.operator <<(t2);

t1.operator >>(t2);

}

Tasks # 02

#include<iostream>

#include<string>

#include<cmath>

using namespace std;

class time{

private:

int hr,min ,sec;

public:

void set\_time(int a ,int b,int c)

{

a = hr;

b = min;

c = sec;

}

bool operator + (const time & t2)

{

hr = hr + t2.hr;

if (hr >=12)

{

hr = hr -12;

}

min = min + t2 .min;

if (min >= 60)

{

hr = hr + 1;

min = min - 60;

}

sec = sec + t2 .sec;

if (sec >= 60)

{

min = min +1 ;

sec = sec -60;

}

}

bool operator - (const time & t2)

{

hr = hr - t2.hr;

min = min - t2 .min;

sec = sec - t2 .sec;

}

bool operator > (const time & t2)

{

if (hr > t2.hr && min > t2.min && sec > t2.sec)

return true;

else

return false;

}

bool operator < (const time & t2)

{

if (hr < t2.hr && min < t2.min && sec < t2.sec)

return true;

else

return false;

}

bool operator >= (const time & t2)

{

int m,n;

m = hr \* 3600 + min \* 60 + sec;

n = t2.hr \* 3600 + t2.min \* 60 + t2.sec;

if (hr = t2.hr && min = t2.min && sec = t2.sec)

return true;

else if(m>n)

return true;

else

return false;

}

bool operator <= (const time & t2)

{

int m,n;

m = hr \* 3600 + min \* 60 + sec;

n = t2.hr \* 3600 + t2.min \* 60 + t2.sec;

if (hr = t2.hr && min = t2.min && sec = t2.sec)

return true;

else if(m<n)

return true;

else

return false;

}

bool operator != (const time & t2)

{

if (hr = t2.hr && min = t2.min && sec = t2.sec)

return false;

else

return true;

}

time operator ++(int)

{

//post form

t1.hr = hr ++;

t1.min = min ++;

t1.sec = sec ++;

return t1;

}

bool operator ++(const time &t2)

{

t2.hr = ++ hr;

t2.min = ++ min;

t2.sec = ++ sec;

return t2;

}

time operator --(int)

{

t1.hr = hr --;

t1.min = min --;

t1.sec = sec --;

return t1;

}

bool operator --(const time &t2)

{

t2.hr = -- hr;

t2.min = -- min;

t2.sec = -- sec;

return t2;

}

int main()

{

time t1 ,t2;

t1.set\_time(1,12,32);

t1.operator <<(t2);

t1.operator >>(t2);

t1.operator +(t2);

t1.operator -(t2);

t1.operator >(t2);

t1.operator <(t2);

t1.operator <=(t2);

t1.operator >=(t2);

t1.operator !=(t2);

t1.operator ++();

t1.operator ++(t2);

t1.operator --();

t1.operator --(t2);

}