(1) Is the following code legal? If not, explain what errors it contains. If legal, explain exactly what it does.
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
    int x = 1, y = 6;
    vector<int> v{8,5,1,4,6,3,7};  //Put stuff into v

    auto n = count_if(v.begin(), v.end(), [=](int i) { return i > y-x; });
    cout << n << endl;
}

ANS:

The code is legal. The vector v is initialized then the count_if algorithm is applied to v. In particular a lambda function is used as the predicate for count_if. The lambda function determines if the elements in v are greater than y-x (where x,y are local variables captured by the lambda function). Since y-x = 5 the code prints out how many elements of v are greater than 5 (there are 3 such elements).

(2) Is the following code legal? If not, explain what errors it contains. If legal, explain exactly what it does.
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
    int x = 1, y = 6;
    vector<int> v{8,5,1,4,6,3,7};  //Put stuff into v

    auto n = count_if(v.begin(), v.end(), [=](int i) { return i > y-x; });
    cout << n << endl;
}

ANS:

The code is legal. The vector v is initialized then the count_if algorithm is applied to v. In particular a lambda function is used as the predicate for count_if. The lambda function determines if the elements in v are greater than y-x (where x,y are local variables captured by the lambda function). Since y-x = 5 the code prints out how many elements of v are greater than 5 (there are 3 such elements).
v = move(v);

for(auto i: v)
    cout << i << " ";
}

ANS:
The code is legal if dangerous. A vector v is initialized and then the elements of v are printed out. The move assignment operator is called on v itself. This means that the data from v is copied to itself and then v is deleted. That is why nothing is printed out after the move assignment. In general you don’t want to do this.

(3) Write a class Point that represents a point on the screen. Hence it has an x and y coordinate (which are integers). You should write a class definition and the following functions (you must decide if they are members, friends, or global).

(a) A constructor which is passed two integers, consider whether to use default values.
(b) An assignment operator.
(c) A translate function which is passed an integer and changes both coordinates by that amount.
(d) A rotate function which is passed an angle and rotates the point by that amount. If the original point is (x,y) and the angle of rotation is θ, then the new coordinates (x',y') are given by the formula:

\[ x' = x \cos \theta - y \sin \theta, \quad y' = x \sin \theta + y \cos \theta \]

(e) A distance function that gives the distance between two points. The formula for the distance between (x,y) and (x',y') is:

\[ \sqrt{(x - x')^2 + (y - y')^2} \]

(f) A function that determines if a point is on a given line. The information for the line would be passed in as a slope and a specific point on the line. Hint: use the point-slope formula for a line.
(g) Functions to access the x and y coordinates.
(h) Functions to modify the x and y coordinates.
(i) A function to convert the point \((x, y)\) into polar coordinates \((r, \theta)\). The conversion equations are:

\[ r = \sqrt{x^2 + y^2}, \quad \theta = \tan^{-1}(y/x) \]

(j) Input and output operators.

Write a main program that tests each of these functions. You must turn in your code, including your main program by email.

ANS:
//File point.h
#include <iostream>
#include <cmath>
using namespace std;

const double PI = 3.1415926535;
class Point {
    int x, y;

public:
    Point(int a=0, int b=0):x(a), y(b) {}
    Point &operator=(const Point &p); //Assignment

    void setX(int a) { x = a; }
    void setY(int b) { x = b; }
    int getX() const { return x; }
    int getY() const { return y; }

    void translate(int a);
    void rotate(double arg);
    void polar(double &r, double &theta);
};

double distance(const Point &p1, const Point &p2)
{
    double x1 = p1.getX(), y1 = p1.getY();
    double x2 = p2.getX(), y2 = p2.getY();
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
}

istream &operator>>(istream &is, Point &p)
{
    int x, y;
    is >> x >> y;
p.setX(x); p.setY(y);
    return is;
}

ostream &operator<<(ostream &os, const Point &p)
{
    return os << '(' << p.getX() << ',' << p.getY() << ');
}

// Check if Point p is on line containing Point p0 and having slope m
bool is_on_line(const Point &p, const Point &p0, double m)
{
    double x = p.getX(), y = p.getY();
    double x0 = p0.getX(), y0 = p0.getY();
    return (y - y0) == m*(x - x0);
}

#include "point.h"

using namespace std;
void Point::translate(int a)  
{  
    x += a; y += a; 
}

void Point::rotate(double arg)  
{  
    int oldx = x, oldy = y;  
    x = (int)(oldx*cos(arg) - oldy*sin(arg));  
    y = (int)(oldx*sin(arg) + oldy*cos(arg)); 
}

void Point::polar(double &r, double &theta)  
{  
    r = sqrt(x*x+y*y);  
    theta = atan(y/x); 
}

Point &Point::operator=(const Point &p)  
{  
    x = p.x;  
    y = p.y;  
    return *this; 
}

//Test program, in its own file
#include <iostream>
#include "point.h"
using namespace std;

int main()  
{  
    Point p1;  
    cout << "Default point = " << p1 << endl;

    Point p2(5,-4);  
    cout << "\nPoint = " << p2 << endl;

//Check assignment  
    p1 = p2;  
    cout << "\nAssignment: " << p1 << endl;

//Check translate function  
    cout << "\nOld point = " << p1 << endl;  
    p1.translate(4);  
    cout << "\nTranslated point = " << p1 << endl;

//Check rotation  
    Point p3(1,0);  
    cout << "\nOld point = " << p3 << endl;  
    p3.rotate(PI/2);  
}
cout << "Point rotated 90 degrees = " << p3 << endl;

// Check distance function
cout << "Distance between points " << p1 << " and " << p2 << endl;
cout << distance(p1,p2) << endl;

// Check is_on_line function for cases point is or isn't on line
Point p4(1,-1), p5(0,-1), fixed(2,1);
double m = 2.0; // Slope of line

cout << "Point " << p4 << " is ";
if( !is_on_line(p4,fixed,m) )
    cout << "not ";
cout << "on line through point " << fixed << " with slope " << m << endl;

cout << "Point " << p5 << " is ";
if( !is_on_line(p5,fixed,m) )
    cout << "not ";
cout << "on line through point " << fixed << " with slope " << m << endl;

// Check polar coordinate function
Point p6(1,-1);
double r,theta;
p6.polar(r,theta); // r = √2, θ = 7π / 4
cout << “Point: “ << p6 << “ has polar coordinates (“ << r << ‘,’ << theta << ‘)’ << endl;

// Check input and output operators
Point p7;
cout << “Enter a point in form x y, where x and y are integers
”;
cin >> p7;
cout << “The point you entered was “ << p7 << endl;

return 0;