```cpp
//Note: file contains compressed program: cat+mouse.cpp & positions.cpp & positions.h

// positions.cpp begin
#include <iostream>
#include <cmath>
using namespace std;

class Position {
public:
Position() {
    myRadius = 1.0;
    myAngleInRadians = 0.0;
}

Position(float r) {
    myRadius = r;
    myAngleInRadians = 0;
}

Position(float r, float thetaInRadians) {
    myRadius = r;
    myAngleInRadians = thetaInRadians;
}

void SetAbsolutePosition(float r, float thetaInRadians) {
    myRadius = r;
    myAngleInRadians = thetaInRadians;
}

void IncrementPosition(float rChange, float angularDistChange) {
    if (rChange != 0 && angularDistChange != 0){
        return;
    }
    myRadius += rChange;

    if (this->IsAtStatue()) {
        myRadius = 1.0;
    }

    float angleChange = angularDistChange/myRadius;
    myAngleInRadians += angleChange;
}

void Print() const {
    cout << "Radius: " << this->myRadius << endl;
    cout << "Angle: " << this->myAngleInRadians << endl;
}

bool Sees(Position mouse) const {
    if (this->myRadius * cos(this->myAngleInRadians - mouse.myAngleInRadians) >= 1.0) {
        return true;
    }
    return false;
}

bool IsAtStatue() const {
    if (this->myRadius <= 1.0) {
        return true;
    }
    return false;
}

bool IsBetween(Position pos1, Position pos2) const {
    float mouseAng = myAngleInRadians;
    if (cos(mouseAng - pos1.myAngleInRadians) >= cos(pos2.myAngleInRadians -
    pos1.myAngleInRadians)
        && cos(pos2.myAngleInRadians - mouseAng) >= cos(pos2.myAngleInRadians -
        pos1.myAngleInRadians)) {
        return true;
    }
    return false;
}

void tests() {

    Position test1;
    Position test2 (7.6);
    Position test3 (3.20, 6.309);
    cout << "Testing constructor, expected (1,0)\n";
    test1.Print();
    cout << "\nTesting overloaded constructor1, expected (rand,7.6)\n";
    test2.Print();
    cout << "\nTesting overloaded constructor2, expected (3.20,6.309)\n";
    test3.Print();

    cout << "\nTesting SetAbsolutePosition, expected (2.22, 4.67)\n";
    test3.SetAbsolutePosition(2.22, 4.67);
    test3.Print();

    cout << "\nTesting IncrementPosition, expected (1,-4)\n";
    test1.IncrementPosition(0, -4);
    test1.Print();

    cout << "\nexpected (3.3,-4)\n";
    test1.IncrementPosition(2.3, 0);
    test1.Print();

    cout << "\nTesting Sees \n";
    for (int i = 0; i < 5; i++) {
        float a,b,c;
        a = rand();
        b = rand();
        c = rand();
        Position testCat (a,b);
        Position testMouse (c);

        float expected = (a*cos(b - c) >= 1.0);
        cout << "expected " << expected << endl;
        cout << testCat.Sees(testMouse) << endl;
    }

    cout << "\nTesting IsBetween \n";
    Position testCat (1.0,1.0);
    cout << "expected 0\n";
    cout << testCat.IsBetween(0.0,0.9);
    cout << "\nexpected 1\n";
    cout << testCat.IsBetween(0.9,1.1);
    cout << "\nexpected 1\n";
    cout << testCat.IsBetween(6.78,7.58);

    cout << "\n\nTesting IsAtStatue\n";
    cout << "expected 1\n";
    cout << testCat.IsAtStatue();
    testCat.IncrementPosition(4.3, 0);
    cout << "\nexpected 0\n";
    cout << testCat.IsAtStatue();
}
private:
    float myRadius;

    float myAngleInRadians;
};

// positions.cpp end

// cat+mouse.cpp begin
// #include positions.h
// #include <iostream>
// #include <cmath>
// using namespace std;
void GetPositions(Position& cat, Position& mouse)
{
    float catR;
    float catAng;
    float mouseAng;

    cout << "Please enter cat radius.\n";
    cin >> catR;

    cout << "Please enter cat angle (degrees)." << endl;
    cin >> catAng;

    cout << "Please enter mouse angle (degrees)." << endl;
    cin >> mouseAng;

    catAng = catAng*M_PI/180;
    mouseAng = mouseAng*M_PI/180;

    cat = Position(catR,catAng);
    mouse = Position(1,mouseAng);
}

/**
 * You define the RunChase function.
 * Given initialized cat and mouse positions,
 * it should simulate the cat chasing the mouse, printing the
 * result of each movement of cat and mouse.  Either the cat will
 * catch the mouse, or 30 time units will go by and the cat will
 * give up.
 */
void RunChase(Position& cat, Position& mouse)
{
    Position newCatPosition = cat;
    Position oldCatPosition = cat;

    int counter = 0;
    int const maxTime = 30;
    bool caught = false;

    if (mouse.IsBetween(cat,cat)) {
        cout << "\nMouse caught immediately\n";
        caught = true;
    }

    while (counter < maxTime && caught == false) {

        if (oldCatPosition.Sees(mouse)) {
            newCatPosition.IncrementPosition(-1.0, 0.0);
        } else {
            newCatPosition.IncrementPosition(0.0, 1.25);
        }

        counter++;

        if (mouse.IsBetween(oldCatPosition, newCatPosition)
            && newCatPosition.IsAtStatue()) {
            caught = true;
            cout << "\nMouse caught at time: " << counter << endl;
        } else {
            mouse.IncrementPosition(0.0, 1);
        }
        cout << endl << counter;
        cout << "\nMouse\n";
        mouse.Print();
        cout << "\nCat\n";
        newCatPosition.Print();
        oldCatPosition = newCatPosition;
    }

    if (!caught) {
        cout << "Cat wandered off.\n";
    }
}

//

int main()
{
    Position cat, mouse;
    GetPositions(cat, mouse);
    RunChase(cat, mouse);
    return 0;
}
```