Problem 1

Define the class **Segment** representing segment $[A, B]$ of the Cartesian number axis

```
class Segment {
    double A,B;
public:
    Segment(double A, double B) : A(A), B(B) { }
    // ...
};
```

Then define methods and functions, so that for segment seg and number d of type **double**

- the value of d*seg or seg*d is a segment which is equal to seg scaled by d (i.e., coordinates of its beginning and end are equal to d*A and d*B, where A and B are coordinates of beginning and end of segment seg);
- the value of seg/d is a segment seg scaled by $\frac{1}{d}$ (segment seg 'divided' by d);
- the value of seg+d or d+seg is a segment shifted (translated) by d to the right;
- the value of seg-d is a segment shifted (translated) by d to the left;
- the value of seg1+seg2 is the smallest segment containing both seg1 and seg2;
- the value of seg(d) is **true** if, and only if, d belongs to seg.

Also, overload **operator<<** so the following **main** function

download *Segment.cpp*

```
int main() {
    using std::cout; using std::endl;

    Segment seg{2,3}, s = 1 + 2*((seg-2)/2+seg)/3;

    cout << s << endl << std::boolalpha;
    for (double x = 0.5; x < 4; x += 1)
        cout << "x=" << x << ": " << s(x) << endl;
}
```

prints something like

```
[1,3]
x=0.5: false
x=1.5: true
x=2.5: true
x=3.5: false
```