```
public static class LineClass
{
    public static IEnumerable<Func<int, int>> LineProvider(
         this IEnumerable<int>? slopes, IEnumerable<int>? yIntercepts)
    {
        IEnumerable<Func<int, int>> LineProvider_Aux()
        {
            using (var it1 = slopes.GetEnumerator())
            {
                using (var it2 = yIntercepts.GetEnumerator())
                {
                    while (it1.MoveNext())
                    {
                        if (it2.MoveNext())
                        {
                            yield return (x) => it1.Current * x + it2.Current;
                        }
                        else yield return (x) => it1.Current * x;
                    }

                    if (it2.MoveNext()) throw new ArgumentException("o");
                }
            }
        }
        if (slopes == null || yIntercepts == null)
          throw new ArgumentNullException("a");
        return LineProvider_Aux();
    }
}

public class LineTest
{
    IEnumerable<int> Resolve(IEnumerable<Func<int, int>> functions, int x)
    {
        foreach (var func in functions)
        {
            yield return func(x);
        }
    }

    [Test]
    public void NoSlopes() =>
        Assert.That(() => Array.Empty<int>().LineProvider(new[] {12}).ToArray(),
         Throws.TypeOf<ArgumentException>());

    [Test]
    public void MoreSlopes()
    {
        Func<int,int> a = (x) => x + 5;
        Func<int, int> b = (x) => 4;
        Func<int, int> c = (x) => -7 * x;
        var ris = new[] {1, 0, -7}.LineProvider(new[] {5, 4});
        var expected_result = new[] {a, b, c};
        Assert.Multiple(() =>
        {
            Assert.That(Resolve(ris,0),Is.EqualTo(Resolve(expected_result,0)));
            Assert.That(Resolve(ris, 100), Is.EqualTo(Resolve(expected_result,
               100)));
            Assert.That(Resolve(ris, 1), Is.EqualTo(Resolve(expected_result, 1)));
            Assert.That(Resolve(ris, -1), Is.EqualTo(Resolve(expected_result, -
               1)));
        });
    }
```

```
    [TestCase(10, 0, 7, new[]{100,110,120,130,140,150,160,170})]
    public void TestInfinite(int whichLine, int minX, int maxX,
          IEnumerable<int> expected)
    {
        IEnumerable<int> genSeq(int x, int a)
        {
            while (true)
            {
                yield return x;
                x += a;
            }
        }

        var slopes = genSeq(0, 1).Take(11);
        var yIntercepts = genSeq(0, 10).Take(11);
        var function = slopes
            .LineProvider(yIntercepts)
            .ElementAt(whichLine);

        Func<int, int> expected_func = (x) => 10 * x + 100;
        var ris = new List<int>();
        for (var i = minX; i <= maxX; i++)
        {
            ris.Add(expected_func(i));
        }

        foreach (var a in ris)
        {
            Console.WriteLine(a);
        }
        Assert.That(ris,Is.EqualTo(expected));
    }
}
```