```
  public static class SmoothingClass
  {
      public static IEnumerable<double> Smooth(this IEnumerable<double> s, int n)
      {
          IEnumerable<double> Smooth_Aux()
          {
              using (var it = s.GetEnumerator())
              {
                  var points = new List<double>();
                  //var points = new double[2 * n + 1];
                  //-the first element-------------------------------------
                  for (int i = 0; i < n+1; i++)
                  {
                      if(!it.MoveNext()) throw new FiniteSourceException();
                      points.Add(it.Current);
                      //points[i] = it.Current;
                  }

                  var sum = points.Sum();
                  var howMany = n + 1;
                  yield return sum / howMany;
                  //-------------------------------------------------------
                  //-elements da 2 a n-------------------------------------
                  for (int i = 0; i < n; i++)
                  {
                      if (!it.MoveNext()) throw new FiniteSourceException();
                      //points[n + 1 + i] = it.Current;
                      points.Add(it.Current);
                      sum += it.Current;
                      howMany++;
                      yield return sum / howMany;
                  }
                  //-------------------------------------------------------
                  //var last = 0;
                  while (true)
                  {
                      if (!it.MoveNext()) throw new FiniteSourceException();
                      sum = sum - points.ElementAt(0) + it.Current;
                      //sum = sum - points[last] + it.Current;
                      points.Remove(0);
                      points.Add(it.Current);
                      //points[last++] = it.Current;
                      yield return sum / howMany;
                  }
              }
          }
          if (null == s) throw new ArgumentNullException(nameof(s));
          if (n <= 0) throw new ArgumentOutOfRangeException(nameof(n));
          return Smooth_Aux();
      }
  }

public class SmoothingTest
  {
      [Test]
      public void FiniteSource_Throws() =>
          Assert.That(() =>
                  new[] {42.0, 49.0, 47.0, 18.0, 19.0, 28.0, 26.0}
                 .Smooth(2).ToArray(),
              Throws.TypeOf<FiniteSourceException>());

      [Test]
      public void NegativeN_Throws()
      {
```

```
        IEnumerable<double> infinite()
        {
            var count = 1.0;
            while (true)
            {
                yield return 0.0 + (count++);
            }
        }
        Assert.That(()=> infinite()
         .Smooth(-1),Throws.TypeOf<ArgumentOutOfRangeException>());
    }
    /*NON TESTATO
    [TestCase(1, new double[] {1, 2, 3, 4, 5, 6},
          new double[] {1.5, 2, 3, 4, 5, 4, 3}, 10)]
    public void InfiniteSequence_ValidArg(int n, double[] sourceSample,
          double[] expectedSample, int howMany)
    {
        IEnumerable<double> InfiniteSeq(IEnumerable<double> s)
        {
            while (true)
            {
                foreach (var num in s)
                {
                    yield return num;
                }
            }
        }
        var result = InfiniteSeq(sourceSample).Smooth(n).Take(howMany);
        var expected =
               expectedSample.Take(howMany).Concat(InfiniteSeq(expectedSample.Skip
         (howMany)));
        Assert.That(result, Is.EqualTo(expected.Take(howMany)));
    }*/
  NON TESTATO
}
```