```
public static class GroupMinClass
{
    public static IEnumerable<int> GroupMin(this IEnumerable<int> individuals,
          int groupSize)
    {
        IEnumerable<int> GroupMin_Aux()
        {
            using (var it = individuals.GetEnumerator())
            {
                while (it.MoveNext())
                {
                    var ris = new List<int>();
                    for (var i = 0; i < groupSize-1; i++)
                    {
                        ris.Add(it.Current);
                        if(!it.MoveNext()) throw new ArgumentException("a");
                    }

                    yield return ris.Min();
                }
            }
        }
        if (null == individuals)
               throw new ArgumentNullException(nameof(individuals));
        if(groupSize <= 0)
               throw new ArgumentOutOfRangeException(nameof(groupSize));
        return GroupMin_Aux();
    }
}

public class GroupMinTest
{
    [TestCase(2,3)]
    public void test1(int size, int groupNumber)
    {
        IEnumerable<int> GenSeq()
        {
            int i = 0;
            for (int j = 0; j < groupNumber * size; j++)
            {
                yield return i;
                i++;
            }
        }
        var ris = GenSeq().GroupMin(size);
        var expectedRis = new List<int>();
        for (int i = 0; i < size+1; i++)
        {
            expectedRis.Add(i*size);
        }
        Assert.That(ris,Is.EqualTo(expectedRis));
    }

    [Test]
    public void test2()
    {
        IEnumerable<int> InfiniteSeq()
        {
            int i = 100;
            while (true)
            {
                yield return i;
                i++;
            }
```

```
        }
        Assert.That(()=> InfiniteSeq().GroupMin(-1),
               Throws.TypeOf<ArgumentOutOfRangeException>());
    }

}
```