```
public static class CountUntilClass
    {
     public static int CountUntil<T>(this IEnumerable<T> seq,
      Predicate<T> matchPredicate,Predicate<T>? exitCondition)
        {
            int CountUntil_Aux()
            {
                using (var it = seq.GetEnumerator())
                {
                    int count = 0;
                    while (it.MoveNext())
                    {
                        if (exitCondition != null && exitCondition(it.Current))
                          return count;
                        if (matchPredicate(it.Current)) count++;
                    }
                    return count;
                }
            }
            if (null == seq || null == matchPredicate)
             throw new ArgumentNullException("A");
            return CountUntil_Aux();
        }
    }

    public class CountUntilTest
    {

        [Test]
        public void stringMatchNumber()
        {
            var seq = "cippalippa".ToCharArray();

            Predicate<char> match = c => char.IsDigit(c);
            Assert.That(seq.CountUntil(match,null),Is.EqualTo(0));
        }

        [Test]
        public void EvenNumber()
        {
            var seq = new[] {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
            Predicate<int> match = i => i % 2 == 0;
            Predicate<int> exit = i => i > 4;
            Assert.That(seq.CountUntil(match, exit), Is.EqualTo(2));
        }

        [Test]
        public void infiniteSeq()
        {
            IEnumerable<int> genSeq()
            {
                var i = 1;
                while (true)
                {
                    yield return i;
                    i++;
                }

            }
            Predicate<int> match = i => i % 2 == 0;
            Predicate<int> exit = i => i > 4;
            Assert.That(genSeq().Take(100).CountUntil(match, exit), Is.EqualTo(2));
        }
    }
```