```csharp
public static class SliceClass
{
    public static IEnumerable<char> Slice(this IEnumerable<string> s,
         int position)
    {
        IEnumerable<char> Slice_Aux()
        {
            using (var it = s.GetEnumerator())
            {
                while (it.MoveNext())
                {
                    if (it.Current is null)
                      throw new ArgumentNullException(nameof(s));

                    var seq = it.Current.ToCharArray();
                    if (seq.Length < position + 1)
                       throw new ArgumentException("a");

                    yield return seq.ElementAt(position-1);
                }
            }
        }
        if (null == s) throw new ArgumentNullException(nameof(s));
        if (position < 0) throw new ArgumentOutOfRangeException(nameof(position));
        return Slice_Aux();
    }
}


public class SliceTest
{
    public IEnumerable<string> genSeq(int howMany, int pos)
    {
        var asciiCode = 32;
        for (var i = 0; i < howMany; i++)
        {
            var sb = new StringBuilder();
            if (asciiCode > 126) asciiCode = 32;

            for (var j = 0; j < pos-1; j++) sb.Append('a');

            sb.Append(Convert.ToChar(asciiCode));
            sb.Append('a');
            asciiCode++;

            yield return sb.ToString();
        }
    }
    [TestCase(5,5)]
    public void Test1(int howMany,int pos)
    {
        IEnumerable<char> genAscii()
        {
            var asciiCode = 32;
            for (var i = 0; i < howMany; i++)
            {
                if (asciiCode > 126) asciiCode = 32;
                yield return Convert.ToChar(asciiCode);
                asciiCode++;
            }
        }

  Assert.That(genSeq(howMany,pos).Slice(pos).Take(100),Is.EqualTo(genAscii().Take(
  100)));
}

[Test]
public void Test2()
{
    IEnumerable<string> infiniteSeq()
    {
        var sb = new StringBuilder();
        yield return "iniziamo!!!!";
        while (true)
        {
            sb.Append("a");
            yield return sb.ToString();
        }
    }
    Assert.That(()=>
           infiniteSeq().Slice(3).Take(34).ToArray(),
           Throws.TypeOf<ArgumentException>());
}
```