Introduction

This lab is to provide some practice working with `HashSet`. For this lab, no skeleton code is provided (but you're free to use code generation tools such as code copilot, carefully checking its suggestions of course!)

The extension (on the second page) does not form part of the assessed content of the lab – it's just intended to provoke some thought and reflection (and investigation if you like!)

Problems

1. Convince yourself by experiment, induction, or otherwise, that in a string of n distinct characters (e.g., $ABCDE$ for $n = 5$) the number of distinct substrings that occur as *factors* of the string, i.e., as blocks of consecutive letters, (including the string itself, and the empty string) is $1 + n(n + 1)/2$.

2. Of course, we usually have strings over a finite *alphabet* and so, in long strings there will be many repeated characters. Write and test a method

   ```
   generateString(char[] c, int n)
   ```

   that generates random strings consisting of n characters from among the characters in `c`. This should work simply by making n random choices of a valid index for `c` and using the character found at that index (remember though that there are good and bad ways to generate strings iteratively!)

3. Do experiments using various values of n and various sizes of arrays of *distinct* characters `c` to investigate how the (expected) number of distinct substrings in a random string compares to the maximum value from question 1. I suggest that you do this by writing (or using) a method that generates the factor of a given string between two indices, and then just use loops and a `HashSet` to collect all of them prior to counting.

4. If `c` does not consist of distinct characters then the random string generation suggested above should result in strings where the characters that occur more frequently in `c` also occur more frequently in the string (this is intentional!) Consider the case where the only characters occurring in `c` are A and B. How does the number of distinct factors of a random string behave as we change the relative proportion of the two letters? What proportion maximises it?

Extension

Consider the following question:

What is the maximum number of distinct factors that a string of length n over a two-letter alphabet can have?

- How could we begin to gather data for this question?
- For how large a value of n do you think we would be able to *know* that we had the final answer? (Assuming of course that we haven't proven a theorem)
- What if we change the size of the alphabet?