## **Download and Extract**

An initial setup of files is provided to you via a shell script: Download potd-q43

Using a terminal, extract the initial files by running the shell script you just downloaded (you will need to navigate to the directory where you saved the file):

```
sh potd-q43.sh
```

Your files for this problem will be in the potd-q43 directory.

## The Problem

We usually restrict the range of hash functions so as to store them in an array. In this POTD, we'll try to study the effect of the range on the performance of the hash function. The performance here shall be measured in terms of the number of collisions that we encounter with the hash function. As we saw in the last POTD, the hash function for a string is sensitive to the order in which the characters in the string are arranged. Here, we'll test the hashing by repeatedly shuffling a string and observing how often the hash of a shuffled string results in a collision. Specifically, given a string str and the range M of the hash function, we'll obtain M permutations of the string. To do so, we'll use the function std::next\_permutation() to permute the string. A sample usage for a string str is as follows:

```
do
{
      // str now contains a permutation of the original string
      // stop permuting after M permutations
} while(std::next_permutation(str.begin(), str.end()));
```

The function std::next\_permutation() keeps supplying permutations untill it reaches the last permutation alphabetically. You must stop permuting beyond M permutations. Your task today is to write a function hash\_goodness() that takes in a string (str) and the range of the hash (M) as the inputs. For M permutations of str, count the number of collisions that result by using the Bernstein hash. Return collisions÷M, which is a measure of how good the hash function is.

## **Example Output:**

```
Goodness of hash Bernstein hash function for "arbitrary" with range=51 is: 0.705882

Goodness of hash Bernstein hash function for "arbitrary" with range=52 is: 0.75

Goodness of hash Bernstein hash function for "arbitrary" with range=53 is: 0.283019

Goodness of hash Bernstein hash function for "arbitrary" with range=54 is: 0.574074

Goodness of hash Bernstein hash function for "arbitrary" with range=55 is: 0.672727

...
```

How does the goodness compare for range values 64, 67 and 100? And for prime numbers in general?

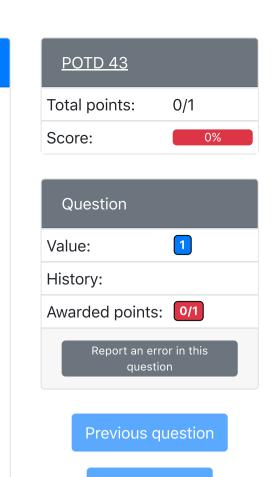
## **Upload Solution**

Drop files here or click to upload.

Only the files listed below will be accepted—others will be ignored.

Files

O Hash.cpp
not uploaded



Next question

