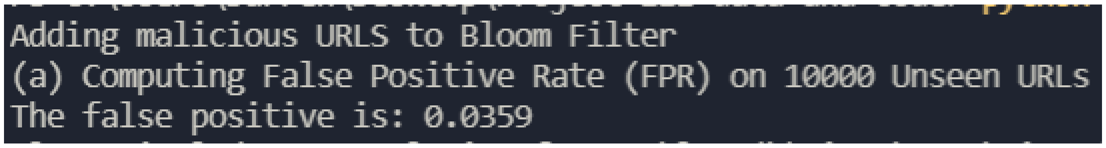
Name: Darren Lee
CruzID: danalee
Email: danalee@ucsc.edu

Project III Report

Instructions to compile/run program
Run the following in the command line terminal:
python bloom_filter.py

Screenshot of program output



```
Adding malicious URLS to Bloom Filter
(a) Computing False Positive Rate (FPR) on 10000 Unseen URLs
The false positive is: 0.0359
```

Problem b)
Let's compare this approach to using a typical Set data structure. Google wants to store 1 million URLs, with each URL taking (on average) 25 bytes. How much space (in MB, 1 MB = 1 million bytes) is required if we store all the elements in a set? How much space (in MB) is required if we store all the elements in a bloom filter with k = 10 hash functions and m = 800,000 buckets? Recall that 1 byte = 8 bits.

Answer to b)
Set:
25 bytes * 1*10^6 URLs= 25 MB

Bloom Filter:
k*m/8=10 hash functions * 800,000 buckets / 8 = 1 MB

Program Description
This program implements a bloom filter using hash functions for quick lookup time to check if an element is added to the filter or not. The add() and contains() functions are implemented by me. Since some elements may map to bits that are already set by the hash functions, there can be false positives reported by the contains() function.

The add() function adds an element to the Bloom filter by setting the bits in the array to true using the hash functions.

The contains() function just checks if the bits for an element are set to check if the element might be added or not. If contains() returns false, the element is definitely not in the set. If contains() return true, the element might be in the set, but there is a chance of
a false positive.