**BDA EXPERIMENT 08**

| **Name:** | **Class/Batch:** |
| --- | --- |
| **Roll No:** | **PRN:** |

**Bloom Filter**

**Program:**

**import math**

**class BloomFilter:**

**def \_\_init\_\_(self, items\_count, bit\_array\_size):**

**self.size = max(bit\_array\_size, 1) # Ensure the bit array size is at least 1**

**self.hash\_count = max(1, self.get\_hash\_count(items\_count, self.size)) # Ensure at least one hash function**

**self.bit\_array = [False] \* self.size**

**def add(self, item):**

**for i in range(self.hash\_count):**

**digest = (hash(item) + i) % self.size**

**self.bit\_array[digest] = True**

**def check(self, item):**

**for i in range(self.hash\_count):**

**digest = (hash(item) + i) % self.size**

**if not self.bit\_array[digest]:**

**return False**

**return True**

**@classmethod**

**def get\_hash\_count(cls, n, m):**

**'''Calculate number of hash functions based on the number of items and size of bit array'''**

**if n == 0 or m == 0:**

**return 0**

**k = (m / n) \* math.log(2)**

**return int(k)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**# Parameters for Bloom filter**

**n = int(input("Enter the expected number of items to add: "))**

**bit\_array\_size = int(input("Enter the size of the bit array: "))**

**# Create Bloom filter**

**bloomf = BloomFilter(n, bit\_array\_size)**

**print("Size of bit array: {}".format(bloomf.size))**

**print("Number of hash functions: {}".format(bloomf.hash\_count))**

**# Adding numbers to Bloom filter**

**print("Enter numbers to add to the Bloom filter (type 'done' to finish):")**

**while True:**

**input\_value = input()**

**if input\_value.lower() == 'done':**

**break**

**try:**

**number = int(input\_value)**

**bloomf.add(number)**

**print(f"Added {number} to the Bloom filter.")**

**except ValueError:**

**print("Please enter a valid number.")**

**# Check presence of numbers**

**print("Enter numbers to check in the Bloom filter (type 'done' to finish):")**

**while True:**

**input\_value = input()**

**if input\_value.lower() == 'done':**

**break**

**try:**

**number = int(input\_value)**

**if bloomf.check(number):**

**print(f"'{number}' may be present (possible positive).")**

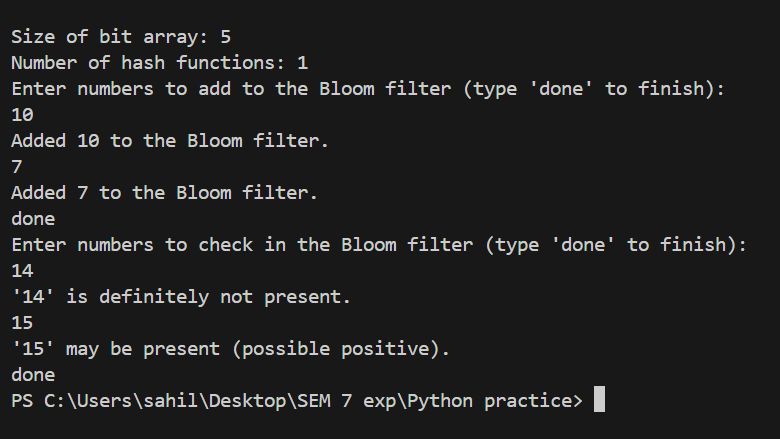
**else:**

**print(f"'{number}' is definitely not present.")**

**except ValueError:**

**print("Please enter a valid number.")**

**Output:**

****