

```
1.Map Reduce:  
#Aim: Count how many times each word appears in input.txt  
  
count = {} # store word-frequency pairs  
  
f = open("input.txt", "r") # open file in read mode  
text = f.read() # read file content  
  
words = text.lower().split() # convert to lowercase & split into words  
  
for word in words:  
    word = word.strip(",") # remove commas and periods  
    if word in count:  
        count[word] += 1 # increase count if word exists  
    else:  
        count[word] = 1 # add new word with count 1  
  
for word, freq in count.items():  
    print(word, ":", freq) # print word and its count  
  
f.close() # close file
```

```
# 2. Bloom Filter Implementation in Python
import hashlib

class BloomFilter:
    def __init__(self, size=20):
        self.size = size
        self.bits = [0] * size

    # Two hash functions (convert number → string for hashing)
    def h1(self, num):
        return int(hashlib.md5(str(num).encode()).hexdigest(), 16) % self.size

    def h2(self, num):
        return int(hashlib.sha1(str(num).encode()).hexdigest(), 16) % self.size

    # Insert number into Bloom Filter
    def insert(self, num):
        i1, i2 = self.h1(num), self.h2(num)
        self.bits[i1] = self.bits[i2] = 1
        print(f"Inserted {num} → bits {i1}, {i2}")

    # Check if number may exist
    def lookup(self, num):
        i1, i2 = self.h1(num), self.h2(num)
        if self.bits[i1] and self.bits[i2]:
            print(f"{num} → may be present")
        else:
            print(f"{num} → not present")

    def show(self):
        print("Bit Array:", self.bits)

# ----- DRIVER CODE -----
bf = BloomFilter(20)

# Insert numeric data
bf.insert(10)
bf.insert(25)
```

```
bf.insert(35)

bf.show()

print("\nLookup:")
for n in [10, 25, 35, 50, 75]:
    bf.lookup(n)
```

```

#3.Flajolet-Martin Algo:

import hashlib

# Function to count trailing zeros in binary
def count_trailing_zeros(num):
    return len(bin(num)) - len(bin(num).rstrip('0'))

# Simple hash function for numeric input
def hash_value(x):
    h = hashlib.sha1(str(x).encode()).hexdigest()
    return int(h[:8], 16) # Take first 8 hex digits

# Flajolet-Martin algorithm
def flajolet_martin(stream):
    R = 0
    for num in stream:
        hv = hash_value(num)
        r = count_trailing_zeros(hv)
        R = max(R, r)
    return 2 ** R # Estimated distinct elements

# ----- DRIVER CODE -----
data_stream = [10, 20, 10, 30, 40, 20, 50, 60, 70, 30, 80]

print("Data Stream:", data_stream)
print("True Unique Elements:", len(set(data_stream)))
print("Estimated Unique Elements (Flajolet-Martin):",
flajolet_martin(data_stream))

```