Assignment 10: Bloom Filters

The Internet is very large, very fast, and filled with degeneracy. The proliferation of foreign persons who not only do not speak English, they use it in corrupt and deviant ways. The untermenschen spend their days sending each other cat videos. Your hero, E.B. decided that in his country of Pacifica that a more neutral *newspeak* was required to keep the people content, pure, and from thinking too much. But how do you process so many words as they flow into your little country at 10Gbits/second? The answer that comes to your brilliant and pure mind is that you use a *Bloom filter*.

A Bloom filter is a hash table, where the entries in the hash table are simply single bits. Consider h(*text*) = k, then if $B_k = 0$ then the entry is definitely missing; if $B_k = 1$ then the entry may be present. The latter is called a false positive, and the false positive rate depends on the size of the Bloom filter and the number of texts that hash to the same position (hash collisions).

You assign your programming minions to take every *potentially offensive* word that they can find in an English dictionary and hash them into the Bloom filter. That is, for each word the corresponding bit in the filter is set to 1.

You will make *two* such Bloom filters. Why? to reduce the chance of a *false positive*. Each Bloom filter should be relatively large (thousands of entries), and use a different hash function. This is easily accomplished by giving our hash function a different input block.

When presented with a stream of text, it is first passed to the first layer Bloom filter. If the Bloom filter rejects any words then the person responsible is innocent of a *thoughtcrime*. But if the word has a corresponding bit set in the Bloom filter, it is likely that they are *guilty* of a thoughtcrime, and that is very ungood indeed. To make certain, we must consult the *hash table* and if the work is there as a *proscribed* word then they will be placed into the care of *Miniluv* and is sent off to *joycamp*. If the word passed both Bloom filters, and is not a forbidden word then the hash table will provide a translation that will replace offensive, insensitive, and otherwise dangerous words with new approved words. The advantage is that your government can augment this list at any time via the *Minitrue*.

Minitrue can also add words from the Bloom filter that you, in your wisdom, have determined are not wholesome for your people to use. There are three cases to consider:

1. Words that are approved, these will not appear in the Bloom filter.

2. Words that should be replaced, which will have amapping from the old word to the new approved word, and

3. Words where no mapping to new approved words means that it's off to *joycamp*.

You will use the *bit vector* data structure that you developed for counting bit project to implement your Bloom filter.

```c
# ifndef NIL
# define NIL (void *) 0
# endif
# ifndef _BF_H
# define _BF_H
# include <stdint.h>
# include <stdlib.h>
# include <stdio.h>

typedef struct bloomF {
    uint8_t *v; // Vector
    uint32_t l; // Length
    uint32_t s[4];  // Salt
} bloomF;

// Each function has its own hash function, determined by the salt.

uint32_t hashBF(bloomF *, char *);

// Create a new Bloom Filter of a given length and hash function.

static inline bloomF *newBF(uint32_t l, uint32_t b[])
{
    // Code
}

// Delete a Bloom filter

static inline void delBF(bloomF *v)
{
    // Code
}

// Return the value of position k in the Bloom filter

static inline uint32_t valBF(bloomF *x, uint32_t k)
```

```
{
    // Code
}

static inline uint32_t lenBF(bloomF *x) { return x->l; }

// Count bits in the Bloom filter

static inline uint32_t countBF(bloomF *b)
{
        // Code
}

// Set an entry in the Bloom filter

static inline void setBF(bloomF *x, char * key)
{
    // Code
}

// Clear an entry in the Bloom filter

static inline void clrBF(bloomF *x, char *key)
{
    // Code
}

// Check membership in the Bloom filter

static inline uint32_t memBF(bloomF *x, char *key)
{
    // Code
}


static inline void printBF(bloomF *x)
{
    // Code
}

# endif
```

bf.h

bf.c is downloadable from the moodle web-site. You may change it to **bf.h** so that it can be included in other programs.

Sample text file to test bloom filter **"gettsburg.txt"** should be used in this project. Please ignore all punctuation marks.