CS5250 Advanced Operating Systems

Pop Quiz 4

Name: ____________Ding Fan________________________

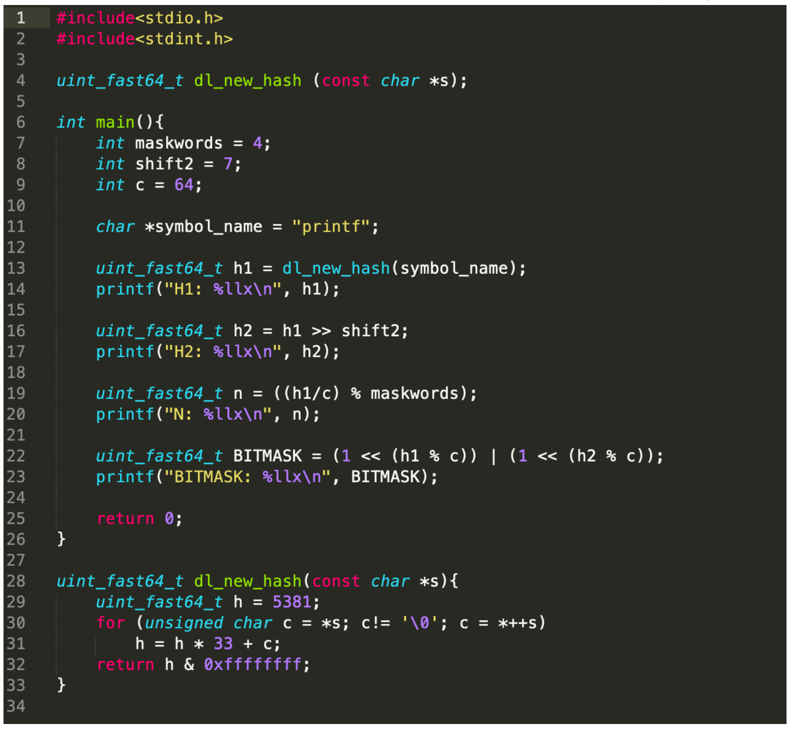
Student Number: __________A0248373X______________

1. Refer to slide 39 of Lecture slide deck 4 “Linking and Loading”.

Assuming that **`maskwords`** = 4, shift2 = 7, and **`C`** = 64, compute **`N`**, and **`BITMASK`** (in hexadecimal) for the string “**`printf`**”. Use the hash function on the same slide. Write down any assumptions that you feel you needed to make.

Solution:

- Because C = 64, this means that the size of one mask word is in 64 bits. We assume that the type of H1, H2, N and BITMASK are unit_fast64_t.
- Because use the same hash function in the slide so the program is designed as following:



```c
#include<stdio.h>
#include<stdint.h>

uint_fast64_t dl_new_hash (const char *s);

int main(){
    int maskwords = 4;
    int shift2 = 7;
    int c = 64;

    char *symbol_name = "printf";

    uint_fast64_t h1 = dl_new_hash(symbol_name);
    printf("H1: %llx\n", h1);

    uint_fast64_t h2 = h1 >> shift2;
    printf("H2: %llx\n", h2);

    uint_fast64_t n = ((h1/c) % maskwords);
    printf("N: %llx\n", n);

    uint_fast64_t BITMASK = (1 << (h1 % c)) | (1 << (h2 % c));
    printf("BITMASK: %llx\n", BITMASK);

    return 0;
}

uint_fast64_t dl_new_hash(const char *s){
    uint_fast64_t h = 5381;
    for (unsigned char c = *s; c!= '\0'; c = *++s)
        h = h * 33 + c;
    return h & 0xffffffff;
}
```

- H1 = dl_new_hash(symbol_name) => H1 = 156B2BB8 (base 16)
- H2 = H1 >> shift2 => H2 = 2AD657 (base 16)
- N = ((H1 / C) % maskwords) => N=2
- BITMASK = (1 << (H1 % C)) | (1 << (H2 % C)) => BITMASK = 180000 (base 16)

2. (I have already mentioned this in the recording, but I want to make sure it "sank in" for you.) For a general Bloom filter using a bit vector of m bits and k hashes, argue why:

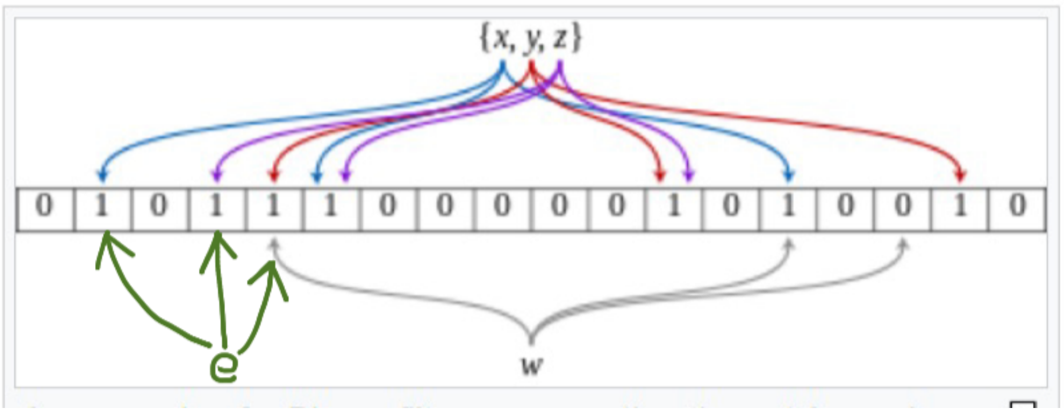
a) If the Bloom filter returns "no, not in the set" for an element e, it must be that e is not in the set.

b) If the Bloom filter returns "yes, may be in the set" for an element e, e may (true positive) or may not (false positive) be in the set. In particular, what would be the worst-case scenario for false positives?

Answer:

a) When we check whether an element e is in a set using Bloom filter, through bloom filter the element e will firstly hash to k different positions in bit vector of m bits. If the Bloom filter returns "no, not in the set" for an element e, it means at least one position of the k hash bits is equal to 0. As we known about set: if an element is in the set, all the k bits must be 1. In this condition (at least one position of the k hash bit is equal to 0), it must be that e is not in the set.

b) If the Bloom filter returns "yes, may be in the set" it means all the k hash bits corresponding to the element e in the bit vector of m bits is equal to 1.
   a. However, we cannot say e is must in this set (e may or may not be in the set).
   b. Refer to the slide of 38 of Lecture slide deck 4 "Linking and Loading" as an example.
   c. x, y, z, is already in the set. If we lookup one of them by the bloom filter, the bloom filter will return "yes, maybe in the set", in this condition it is true positive.
   d. However, if we look up the element e, which is not in the set (showed in the following picture. The bloom filter will return "yes, maybe in the set", because all the k hash bits corresponding to element e are equal to 1. But actually, that these k hash bits equal to 1, is due to a hash collide with the other elements in the set. In this condition it is false positive.
   e. In particular, the worst-case scenario for false positives is that all m bits of the bit vector are equal to 1, because of too many elements in the set. The Bloom filter will always return "yes, may be in the set".
   f.


{x, y, z}
0 1 0 1 1 1 0 0 0 0 0 1 0 1 0 0 1 0
e
w