Justin Marion

Prof Long

CSE 13S

May 4, 2021

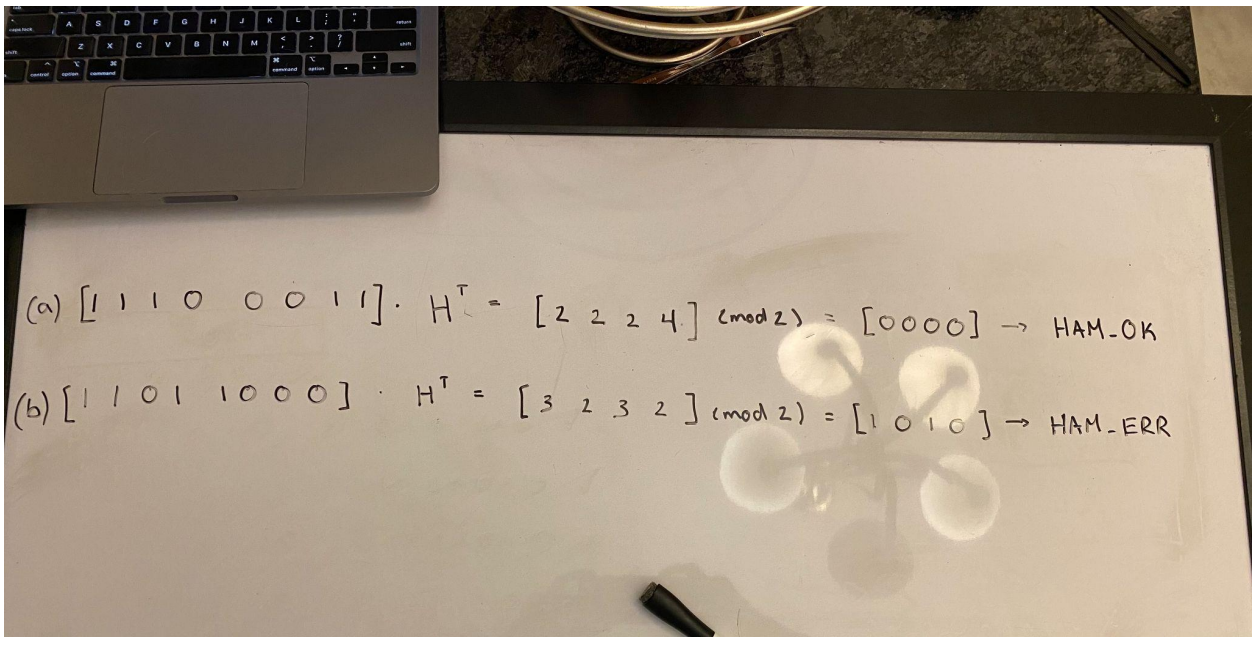
DESIGN.pdf

Pre Lab questions:

1. Look-up Table:

0	HAM_OK
1	4
2	5
3	HAM_ERR
4	6
5	HAM_ERR
6	HAM_ERR
7	3
8	7
9	HAM_ERR
10	HAM_ERR
11	2
12	HAM_ERR
13	1
14	0
15	HAM_ERR

2.



To implement bit vectors I will follow the same principle that prof long showed us in class on Monday: 4/4

```
uint64_t set_bit(uint64_t w, uint8_t v) { return w | ((uint64_t) 0x1 << (v % 64)); }

uint64_t clr_bit(uint64_t w, uint8_t v) { return w & ~((uint64_t) 0x1 << (v % 64)); }

uint64_t get_bit(uint64_t w, uint8_t v) {
    return (w & ((uint64_t) 0x1 << (v % 64))) >> (v % 64);
}
```

Set bits will shift a 64 bit value with only one bit being value 1 to the desired bit that we want to get, then oring the original 64bit and the shifted one to get the desired bit.

Clear does the same as get however it will and the inverse of the shifted 64 bit.

Get bit use the shifted 64 bit and ands the two to simply get the single bit wanted however we need this bit to shift back right so we shift it right the same amount we shifted it left.

Bit Matrix ADT will be nothing special, simple traversing arrays and for loops to multiply arrays.

Finally encode and decode program arguments will carry very similar code from last assignment to open/close and read/write to/from files.