ASGN5

Justin Satriano

Pre-Lab

1)

0	0000	HAM_OK
1	0001	4
2	0010	5
3	0011	HAM_ERR
4	0100	6
5	0101	HAM_ERR
6	0110	HAM_ERR
7	0111	3
8	1000	7
9	1001	HAM_ERR
10	1010	HAM_ERR
11	1011	2
12	1100	HAM_ERR
13	1101	1
14	1110	0
15	1111	HAM_ERR

H^T binary

1	1	1	0
1	1	0	1
1	0	1	1
0	1	1	1
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0

2) 1110 0011 (7 6 5 4 3 2 1 0)

Error in 1st bit

1101
1110
1111
0010
0100
1000

flip 1st bit from 1 to 0

1110 0001

1101 1000

HAM_ERR

1010
0111
0001
0100
1000

error syndrome was 1010 which does not have a binary equivalent

hamming.c

```
uint8_t ham_encode
    m = bm_from_data(msg, 4)
    C = bm_multiply(m, G)
    code = bm_to_code(C)
    return code
```

uint32 m, C, code

create lookup table(array)

```
HAM_STATUS ham_decode
    C = bm_from_data(code)
    e = bm_multiply(C, H^T)
    if e == 0
        return HAM_OK
    else
        L = lookup(e)
    if L == HAM_ERR
        return HAMERR
    else
        flip the L^th bit
        return HAM_CORRECT
```

bv.c

```
Struct Bitvector
    uint32 length
    uint8 *vector

Bitvector *bv_create
    Bitvector *v = (Bitvector*) malloc(sizeof(Bitvector))
    v->length = length
    int bits = length
    if bits % 8 == 0
        v->vector = (uint8*)calloc(bits/8, sizeof(uint8))
    else
        v->vector = (uint8*)calloc(bits/8+1, sizeof(uint8))
    if (!v->vector)
        free(v)
        v = NULL
    return v

void bv_delete
    if (*v && (*v)->vector)
        free((*v)->vector
        free(*v)
        *v = NULL
    return

uint32_t bv_length
    return v->length

void bv_set_bit
    v->vector[i/8] |= (1<<(i%8))

Void bv_clr_bit
    v->vector[i/8] &= ~(1<<(i%8))

uint8_t bv_get_bit
    uint8 result = v->vector[i/8] &(1<<(i%8))
    result >>(i%8)
    return result

void bv_xor_bit
    V->vector[i/8] ^= (bit<<(i%8))

void bv_print
    debug function
```

bm.c

```
Struct Bitmatrix
    uint32 rows
    uint32 cols
    Bitvector *vector

Bitmatrix *bm_create
    Bitmatrix *m = (Bitmatrix*) malloc(sizeof(Bitmatrix))
    m->rows = rows
    m->cols = cols
    m->vector = bv_create(rows * cols)
    return m

void bv_delete
    if (*m && (*m)->vector)
        free((*m)->vector)
        free(*m)
        *m = NULL
    return

uint32_t bm_rows
    return m->rows

uint32_t bm_cols
    return m->cols

void bm_set_bit
    bv_set_bit(m->vector, r*8+c)

void bm_clr_bit
    bv_clr_bit(m->vector, r*8+c)

uint8_t bm_get_bit
    bv_get_bit(m->vector, r*8+c)

Bitmatrix *bm_from_data
    Bitmatrix *m = bm_create(1, length)
    for(uint32 i=0; i<length; i+=1)
        if(byte & (1<<i) == 1)
            bm_set_bit(m->vector, 1, i)
    else
        bm_clr_bit(m->vector, 1, i)

uint8_t bm_to_data
    uint8 value = 0
    for(uint32 i=0; i<M->vector; i+=1)
        value += bm_get_bit(m->vector, 1, i)

Bitmatrix *bm_multiply
    (insert before for loops)
    Bitmatrix *ans = bm_create(A->rows, B->cols)
    uint32 a;
    uint32 b;
    for (uint32 i=0; i<A->rows; i+=1)
        for(uint32 j=0; j<B->cols; i+=1)
            uint32 sum (keeps track of all k values)
            for(uint32 k=0; i<A->cols; i+=1)
                V = bm_get_bit(A->vector, i, k) & bm_get_bit(B->vector, k, j)
                Sum += V
                Sum = Sum%2
                ans->vector = V ^ sum
                            (XOR)

void bm_print
    Debug function
```

encode.c

```
void main()
    getopt with options(h i:o:)
    Create G(generator matrix)(bit matrix)
        G(4,8)
    while(fgetc != EOF)
        byte = something
        l_nibble = lower_nibble(byte)
        u_nibble = upper_nibble(byte)
        code_lower = ham_encode(l_nibble)
        code_upper = ham_encode(u_nibble)
        fputc(code_lower)
        fputc(code_upper)
```

Need to write to output file

close files

decode.c

```
void main()
    get opt with options(h i:o:v)
    Ht = bm_create(8,4)
    while(fgetc != EOF)
        lower = ham_decode(lower_byte)
        upper = ham_decode(upper_byte)
        Packed = Pack_byte(lower, upper)
        fputc(packed)
```

Need to write to output file

close files

Dont forget statistics

Keep track in decoder.c file