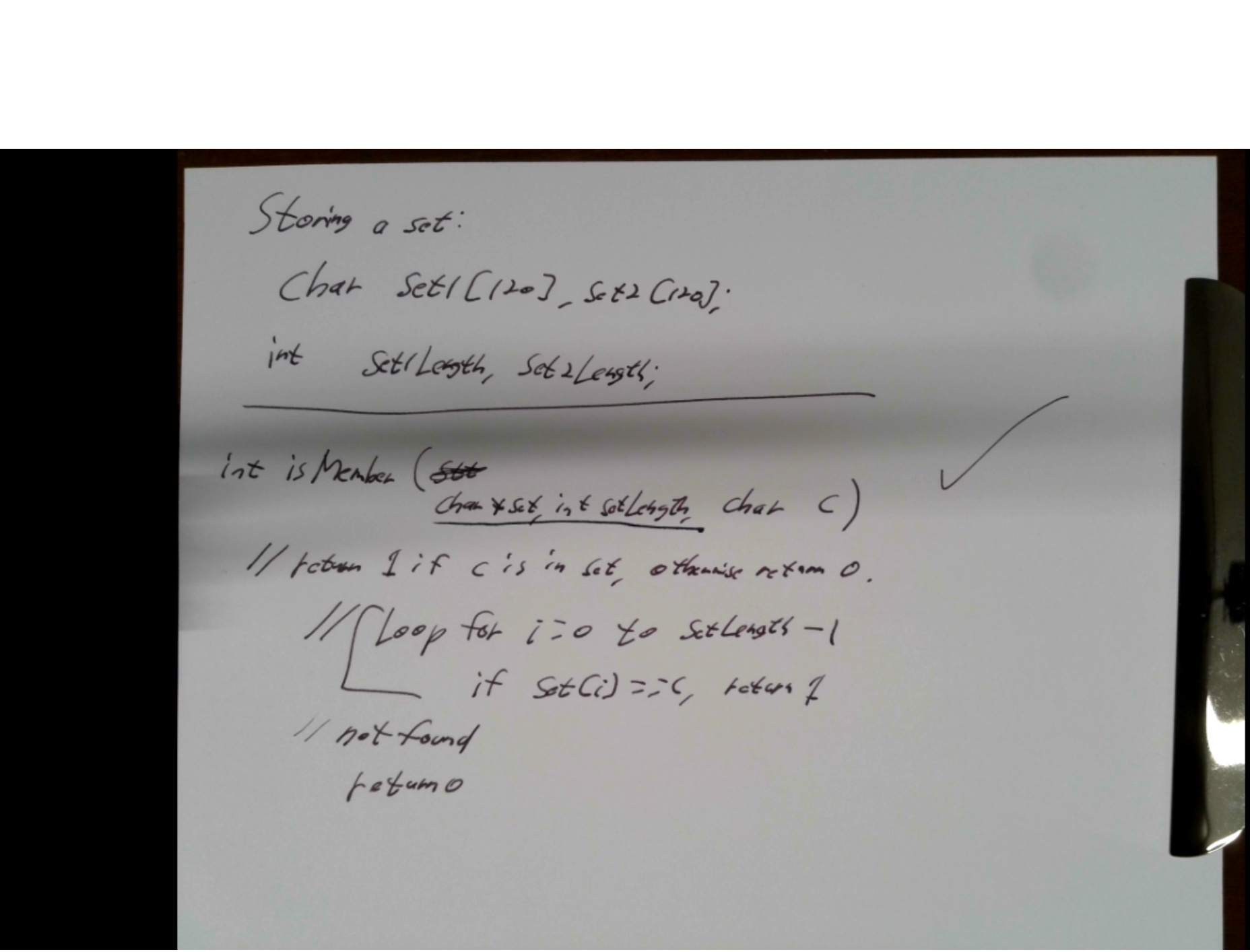
Storing a set:

```
char set1[120], set2[120];
int set1Length, set2Length;
```

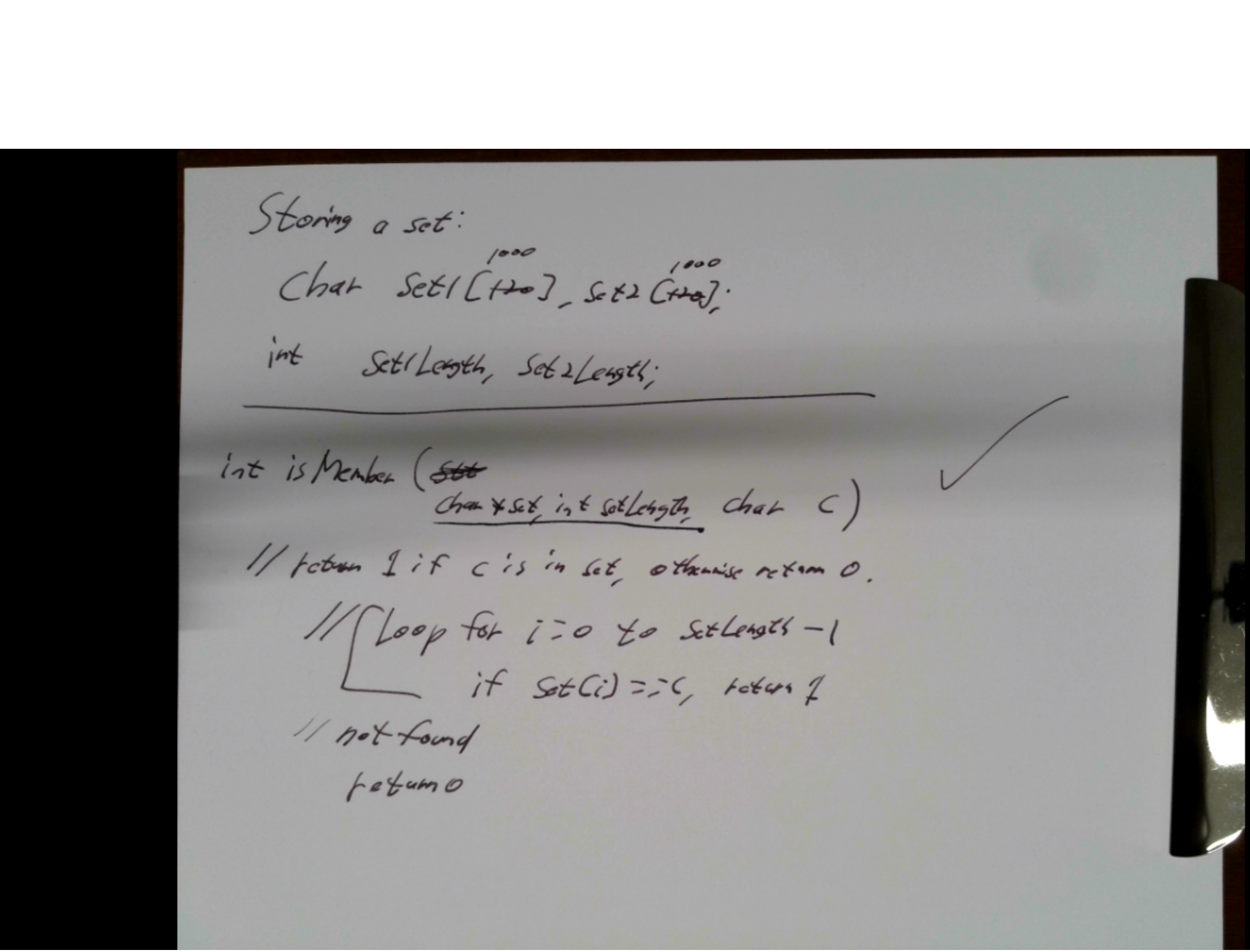
```
int isMember (char *set, int setLength, char c)
// return 1 if c is in set, otherwise return 0.
//[Loop for i:=0 to setLength-1
    if set[i] =:= c, return 1
// not found
  return 0
```



Storing a set:

```
char set1[1000], set2[1000];
int set1Length, set2Length;
```

```
int isMember(char *set, int setLength, char c)
// return 1 if c is in set, otherwise return 0.
//[Loop for i:=0 to setLength-1
    if set[i] == c, return 1
// not found
  return 0
```



```
void Union(S1, L1, S2, L2, S3, L3)

  // S1 ∪ S2 ⟶ S3
  // Set L3 = 0 (so S3 = {})
  for (i=0; i < L1; i++)
        add S1[i] to S3

---

for (i=0; i < L2; i++)
      if((isMember(S3, L3, S2[i]) == 0)
            add S2[i] to S3

---
```

Your program should prompt the user for the elements of the first set (called A): the user enters the elements, followed by $< ENTER >$. Do the same for the second set (B). Then print the following information:

- The elements of A in the usual format: $\{e_1, e_2, ..., e_n\}$
- The elements of B
- The cardinality of A and B
- $A \cup B$
- $A \cap B$
- $A \setminus B$

2 Details

- You should prompt the user for their two input strings.
- Remember, a set cannot contain more than one copy of an element. **If the user enters an element twice, it should only appear in the**

```
add(Set, length, char)
// caller must avoid dups
  Set(length) = char
  length = length+1
```

Applications

Parity

data

parity bit

A = 01000001 | 0

B = 01000010 | 0

C = 01000011 | 1

SEND

- Add up all 8 bits
- calculate value (mod 2)
- ~~add~~ send that value as a 9th bit.

RCV

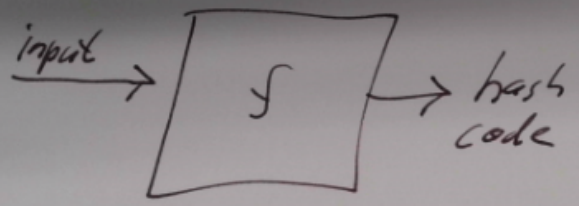
- add all 9 bits
- calculate mod 2
- if = 0, GOOD
  if = 1, BAD

Send A 01000001 0

DRAM with ECC

Error checking & correcting

Hash function



fingerprinting

UPC

11 digits

1 Product Category
5 Manufacturer
5 Item
1 check digit

$C: \left(3x_1 + x_2 + 3x_3 + x_4 + \dots + 3x_{11}\right) \pmod{10}$

if $C \neq 0$, $C = 10 - C$

100% single digit errors
90% transpositions

Crypto

Caesar Cipher: A → C
B → D
C → E
⋮
X → Z
Y → A
Z → B

(Shift Cipher)

Encryption: "Password" is +2

Decryption: -2

Frequency Analysis

A → X

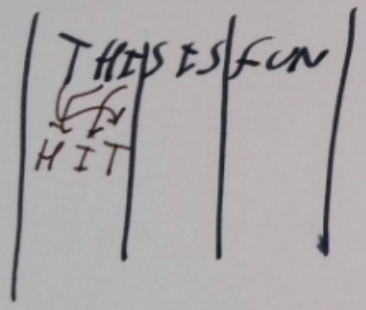
B → C

C → R

D → Q

E → M

Transposition Cipher



1 → 3

2 → 1

3 → 2

A → X

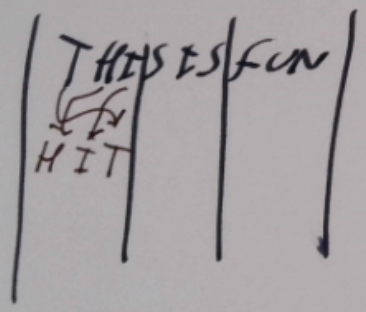
B → C

C → R

D → Q

E → M

Transposition Cipher



1 → 3

2 → 1

3 → 2

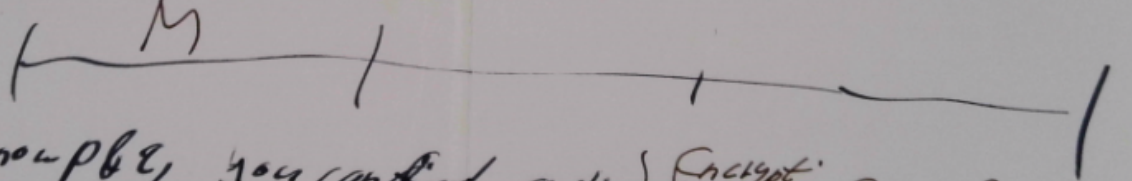
Assymetric Cipher /
public key / private key

RSA

$n = pq$

p, q large primes

e relatively prime to $(p-1)(q-1)$

Message: 

If you know p & q, you can find a #
d so that $de \equiv 1 \mod (p-1)(q-1)$

KEY: (n, e)

Encrypt: $C = M^e \pmod n$

Decrypt: $C^d \pmod n$

$= M$