3/1/22

# * Section 8 - Strings

## 131. Introduction to Strings

* ASCII codes (0-127) = 128
  7 bits - $2^7$ = 128

| | | |
|---|---|---|
| A - 65 | a - 97 | 0 - 48 |
| B - 66 | b - 98 | 1 - 49 |
| C - 67 | c - 99 | 2 - 50 |
| ⋮ | ⋮ | ⋮ |
| Z - 90 | z - 122 | 9 - 57 |

( , ) ? ; + - * . . .

↵ - 10
space - 13
esc - 27

* Unicodes.org
  2 byte = 16 bits

4 x 4 bits
C 0 3 A

* Declare character

```
char temp;   <- 1 byte
temp = 'A';
```

temp
| A 65 |

```
x temp = 'AB';
x temp = A;
x temp = "A";
```

```
printf("%c", temp); => A
printf("%d", temp); => 65

cout << temp; - A
```

* Declare character Array

```
char x[5];
char x[5] = {'A', 'B', 'C', 'D', 'E'};
```

| x | A | B | C | D | E |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |

```
char x[] = {'A', 'B', 'C', 'D', 'E'};
char x[5] = {65, 66, 67, 68, 69};

char x[5] = {'A', 'B'};
```

| x | A | B | 0 | 0 | 0 |
|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 |

* Strings in C/C++

```
char name[10] = {'J', 'o', 'h', 'n', '\0'};
```

Array of characters

name

| J | o | h | n | \0 | o | o | o | o | o |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

String delimeter '\0'
End of string char
NULL char
String terminator

In Java it is known by its length

* Declare string

```
char name[10] = {'J', 'o', 'h', 'n', '\0'};

char name[] = {'J', 'o', 'h', 'n', '\0'};
char name[] = "John";

printf("%s", name);
scanf("%s", name);
gets(name);
```

← Only for One word

John ✓
Taj Mahal ✗
Taj Mahal ✓

132. Finding Length of a string

| s | w | e | l | c | o | m | e | \0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

```
int main(){
    char s[] = "welcome";
    int i;
    for(i=0; s[i]!='\0'; i++){

    }
    printf("Length is %d", i);
    return 0;
}
```

133. Changing case of a string

| A | W | E | L | C | O | M | E | \0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

A - 65 a - 97 97 - 65 = 32

B - 66 b - 98 98 - 66 = 32

:

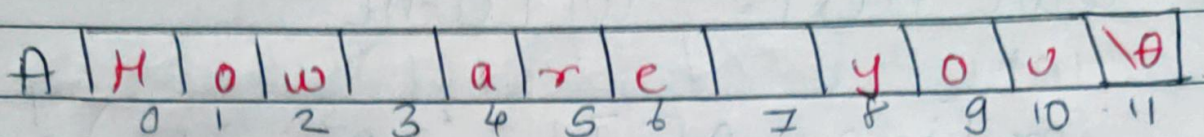
Z - 90 z - 122 122 - 90 = 32

```
int main(){
    char A[] = "WELCOME";
    int i;
    for(i=0; A[i] != '\0'; i++){
        A[i] = A[i] + 32;
    }
    printf("%s", A);
}
```

| A | w | E | l | C | o | m | e | \0 |
|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

```
int main(){                                   //toggling
    char A[] = "wElCome";
    int i;
    for(i=0; A[i] != '\0'; i++){
        if(A[i] >= 65 && A[i] <= 90)
            A[i] += 32
        else if (A[i] >= 'a' && A[i] <= 122)
            A[i] -= 32;
    }
    printf("%s", A);
}
```

134. Counting words and vowels in a String




```
int main(){
    char A[] = "How are you";
    int i, vcount=0, ccount=0;
    for(i=0; A[i]!='\0'; i++){
        if(A[i]=='a' || A[i]=='A' ||
           A[i]=='e' || A[i]=='E' ||
           A[i]=='i' || A[i]=='I' ||
           A[i]=='o' || A[i]=='O' ||
           A[i]=='u' || A[i]=='U')
        {
            vcount++;   // Vowels
        }
        else if((A[i]>=65 && A[i]<=90) ||
                (A[i]>=97 && A[i]<=122))
        {
            ccount++;   //consonants
        }
    printf("%d", vcount);
    printf("%d", ccount);
}
}
```

```
int main(){
    char A[] = "How are you";
    int i, word = 1;
    for(i=0; A[i] != '\0'; i++){
        if(A[i] == ' ' && A[i-1] != ' ')
            word++;
    }
    printf("%d", word);
}
```

For extra whitespace

## 135. Validating a String

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | A | n | i | l | 3 | 2 | 1 | \0 | → Valid |
| A | A | n | i | ? | 3 | 2 | 1 | \0 | → Not valid |
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |

```
int valid(char * name){
    int i;
    for(i=0; name[i] != '\0'; i++){
        if(!(name[i] >= 65 && name[i] <= 90)
        && !(name[i] >= 97 && name[i] <= 122)
        && !(name[i] == 48 .... name[i] == 57))
            return 0;
    }
    return 1;
}
```

```
int main(){
    char *name = "Ani?321";
    if(validate(name)){
        printf("Valid String");
    }
    else{
        printf("Invalid String");
    }
}
```

136 Reversing a String

* First Method

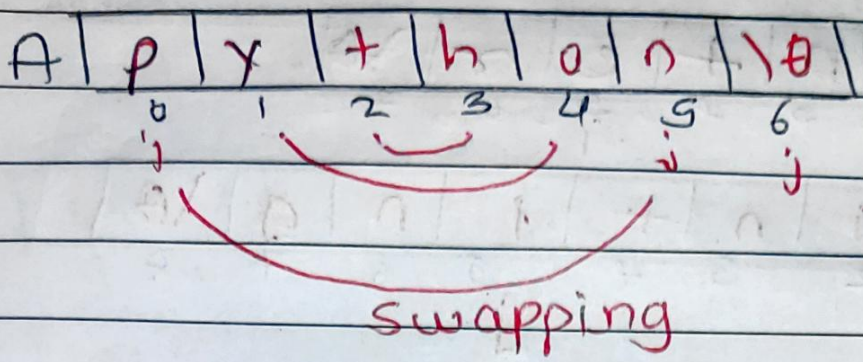
| A | P | y | t | h | o | n | \0 |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

| B | n | o | h | t | y | P | \0 |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

j

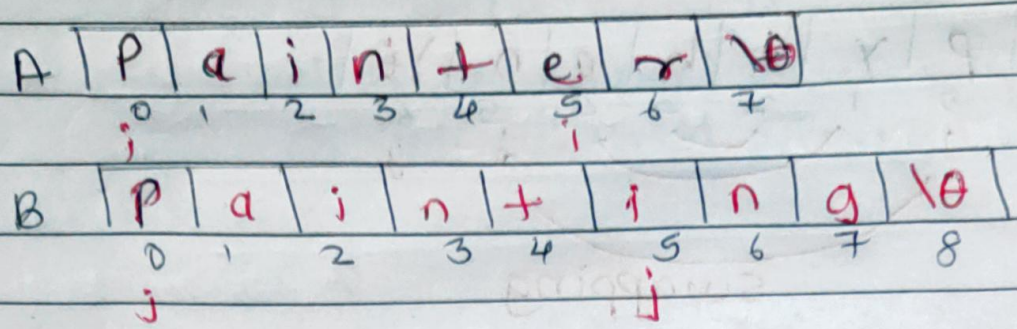
```
int main(){
    char A[] = "python"; // can be changed (mutable)
    // char A = "python"; (Immutable)
    char B[7];
    int i;
    for(i=0; A[i] = '\0'; i++){
        i = i-1;        // i--
        for(j=0; i>=0; i--; j++){
            B[j] = A[i];
        }
        B[j] = '\0';
        printf("%s", B);
    }
}
```

* Second method



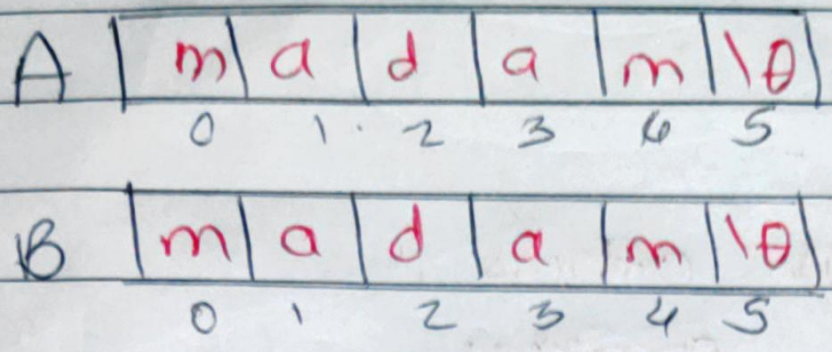

```
int main(){
    char A[] = "python";
    char t;
    int i, j;
    for(j=0; A[j]!='\0'; j++){
        j=j-1;
        for(i=0; i<j; i++, j--){
            t=A[i];
            A[i]=A[j];
            A[j]=t;
        }
    printf("%s", A);
}
}
```

137. Comparing strings and checking Palindrome



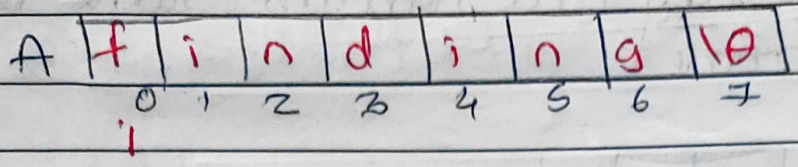

```
int main() {
    char A[] = "Painter";
    char B[] = "Painting";
    int i, j;
    for(i=0; j=0; A[i] != '\0' && B[i] != '\0'; i++, j++)
    {
        if (A[i] != B[j])
            break;
    }
    if (A[i] == B[j])
        printf("Equal");
    else if (A[i] < B[j])
        printf("Smaller");
    else
        printf("Greater");
}
```

* String is Palindrome or not
(Reverse a string and remains same is Palindrome)



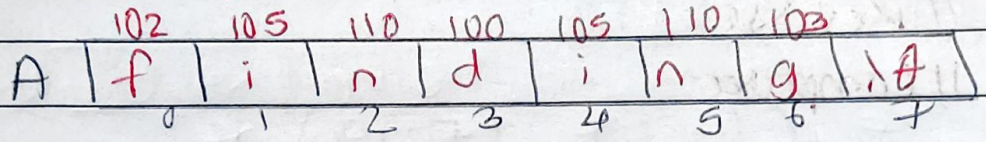

```
int main(){
    char A[] = "madam";
    char B[6];
    // Reverse
    // Compare
```

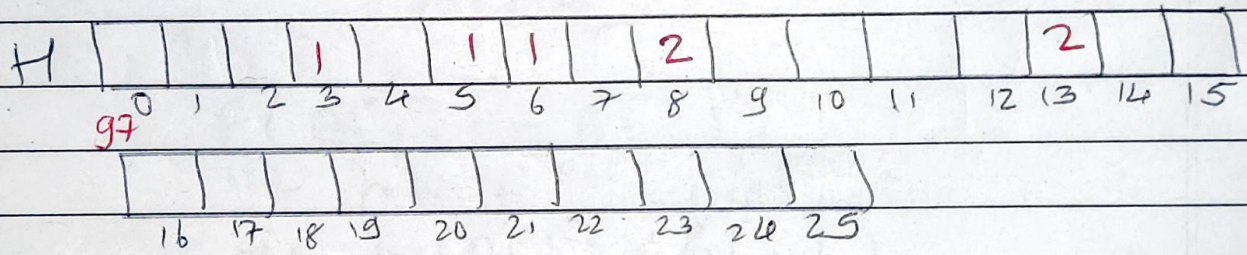
# 138. Finding Duplicates in a String




* First method
  // Comparing with others
  // Same as in array

* Second method
  // using hashtable



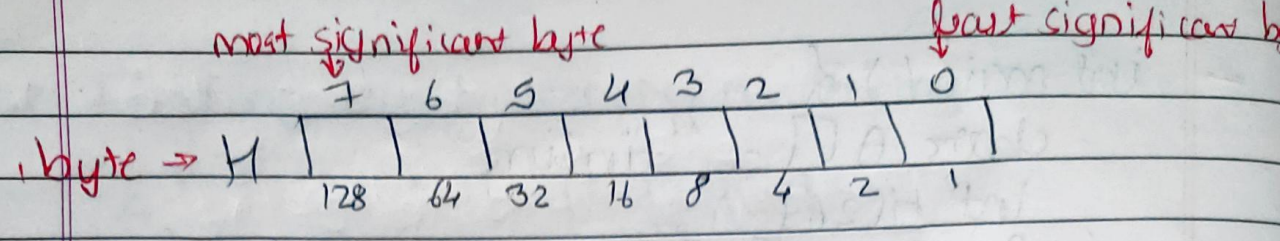

f = 102 - 97 = 5
i = 105 - 97 = 8
n = 110 - 97 = 13
d = 100 - 97 = 3
g = 103 - 97 = 6

```
int main() {
    char A[] = "finding";
    int H[26], i;
    for (i=0; A[i]!='\0'; i++) {
        H[A[i]-97] += 1;
    }
    for (i=0; i<26; i++) {
        if (H[i] > 1) {
            printf("%c:", i+97);
            printf("%d", H[i]);
        }
    }
}
```

Time : O(n)

139. Finding Duplicates in a string using Bitwise Operations.

1. Left shift <<
2. Bits ORing (merging)
3. Bits ANDing (masking)




1) Left shift

H = 1

H = H << 5

10

After left shift

100000

2) AND Operation

a = 10 → 1010

b = 6 → 0110

0010 = 2

a & b = 2

| a | b | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

3) OR Operation

a = 10 → 1010

b = 6 → 0110

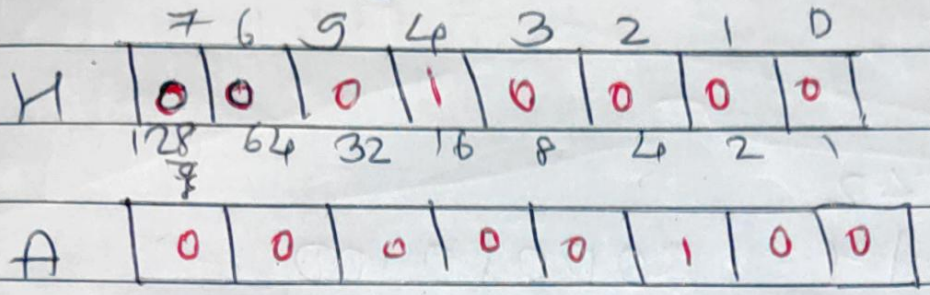
1110 - 14

a | b = 14

| a | b | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Setting the bit ON is called merging
checking whether the bit is ON and OFF is called masking

* Bit masking




Check for 2

a = 1

a = a<<2 a&H

All are zero by performing AND so 2 is OFF

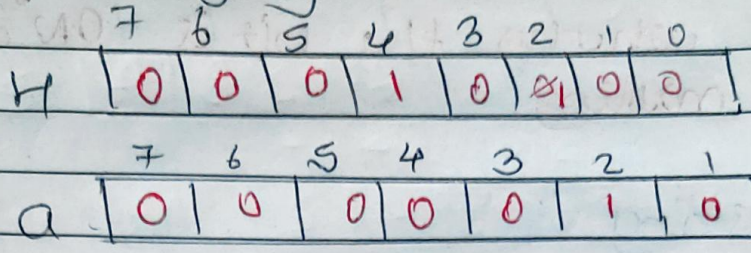
Check for 2

a = 1

a = a<<4 a&H

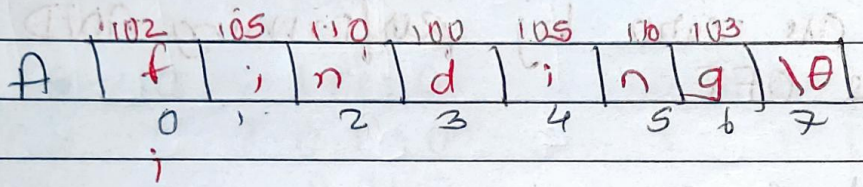
4 is ON

* Bit merging



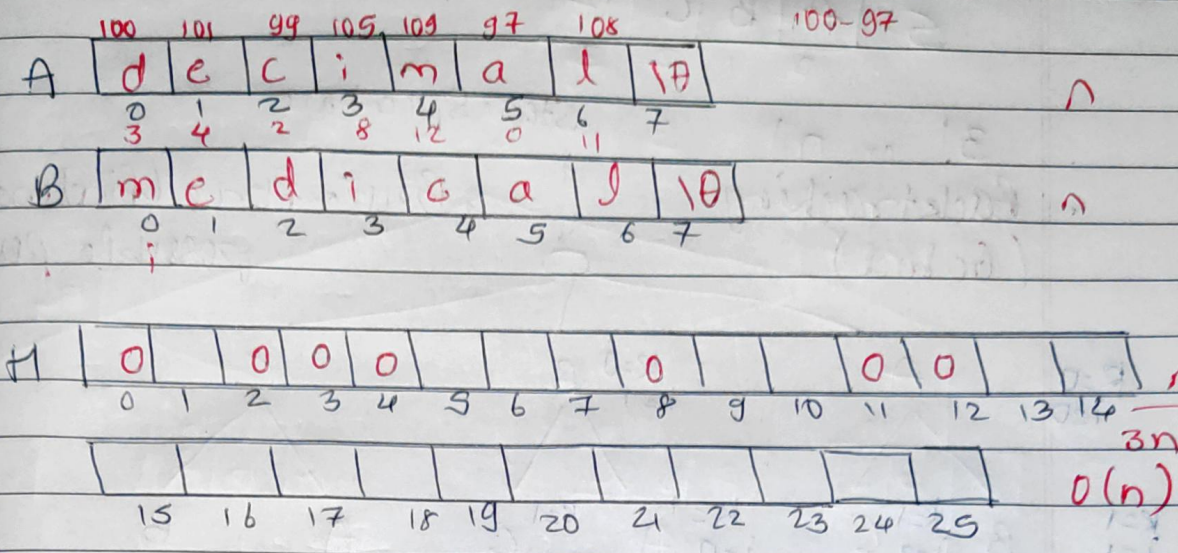

a = 1
a = a<<2
x = a|H  00010100

* Finding Duplicates



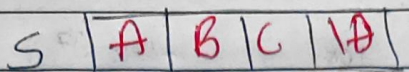

```
int main(){
    char A[] = "finding";
    log int H=0, x=0;
    for(i=0; A[i]!='\0'; i++){
        x=1;
        x = x<<(A[i]-97);
        if((x&H)>0){
            printf("%c is duplicate", A[i]);
        }
        else
            H=x|H;
    }
}
```

140. Checking if 2 Strings are Anagram (distinct letters)




```
int main(){
    char A[] = "decimal";
    char B[] = "medical";
    int i, H[26] = {0};
    for(i=0; A[i]!='\0'; i++){
        H[A[i]-97] += 1;
    }
    for(i=0; B[i]!='\0'; i++){
        H[A[i]-97] -= 1;
        if(H[A[i]-97] < 0){
            printf("Not Anagram");
            break;
        }
    }
    if(B[i]=='\0')
        printf("Its Anagram");
}
```

14. Permutation of a String

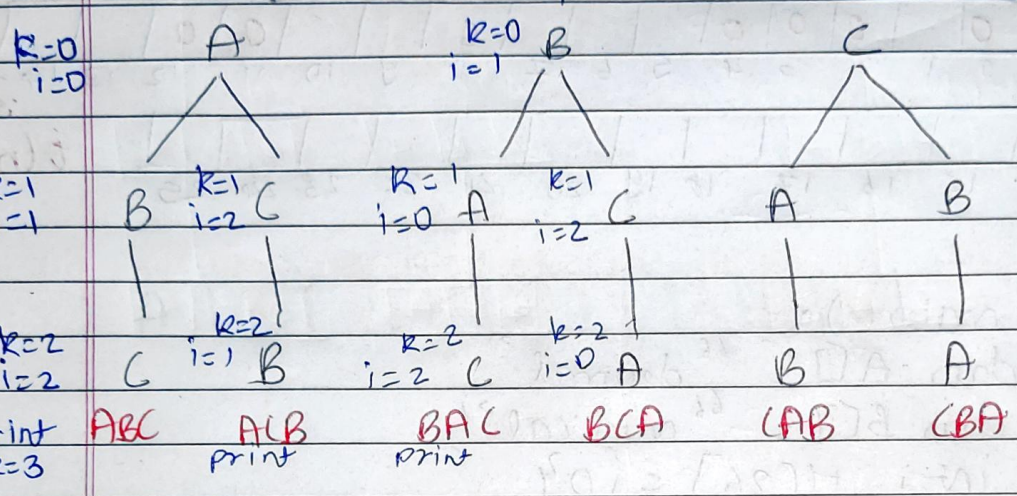



3! or n!

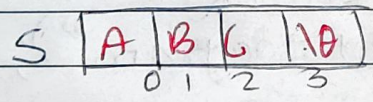
- Backtracking (Go back)

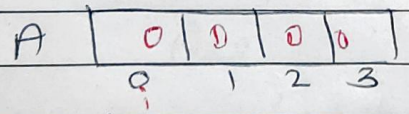
State Spectrale

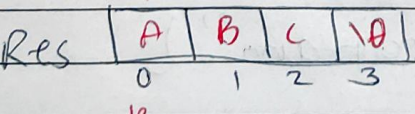
- Brute-force (finding all possible permutation)




- Backtracking
- Brute force => Finding out everything
- Recursion



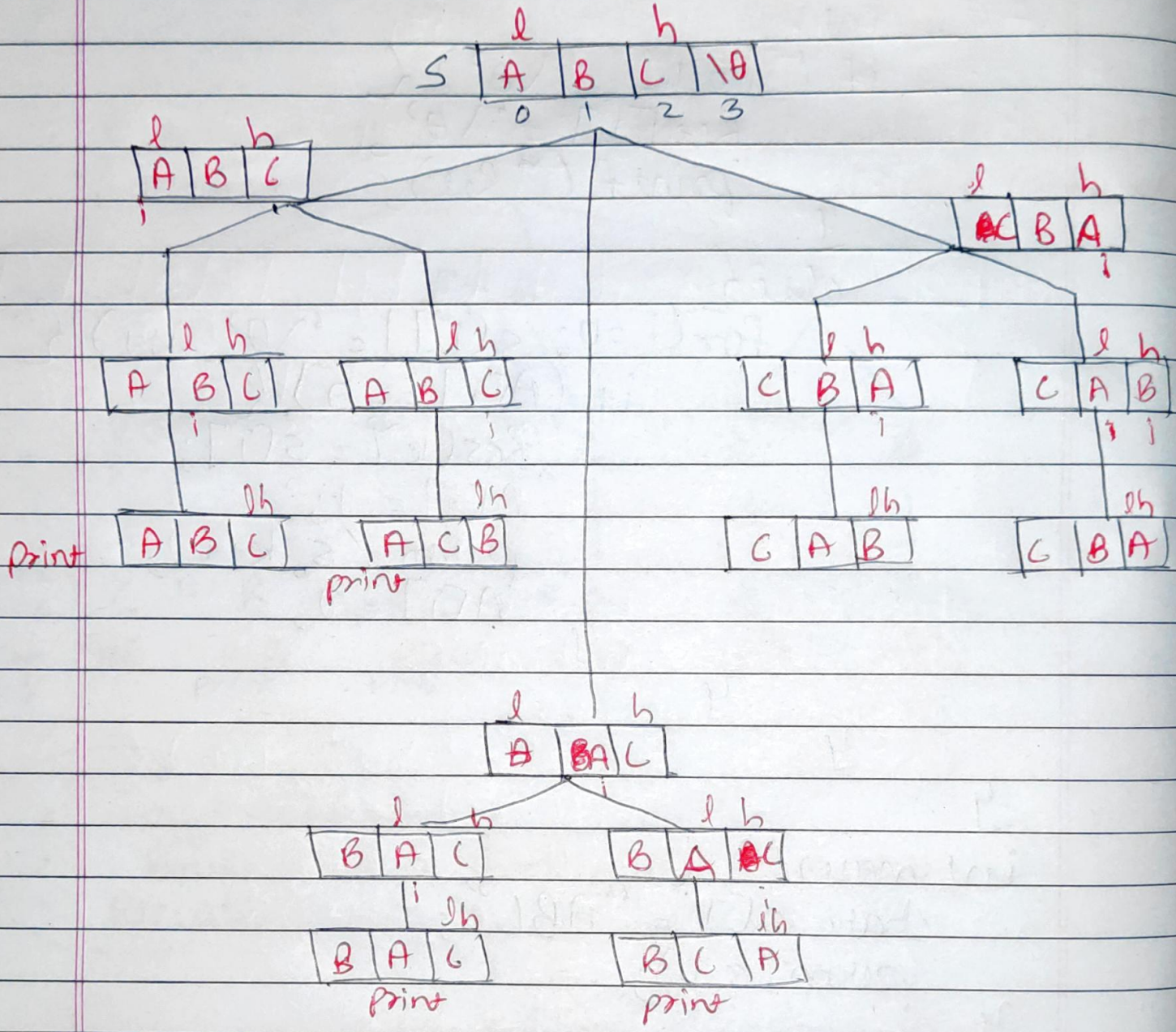



* First Method

```
void perm(char s[], int k) {
    static int A[10] = { 0};
    static char Res[10];
    int i;
    if (s[k] == '\0') {
        Res[k] = '\0';
        printf("%s", Res);
    }
    else {
        for(i=0; s[i] != '\0'; i++) {
            if (A[i] == 0) {
                Res[k] = s[i];
                A[i] = 1;
                perm(s, k+1);
                A[i] = 0;
            }
        }
    }
}
int main() {
    char s[] = "ABC";
    perm(s, 0);
}
```

* Second Method

S | A | B | C | \0



```
void perm (char s[], int l, int h) {
    int i;
    if (l == h) {
        printf (s);
    } else {
        for (i=l; i<=h; i++) {
            swap (s[l], s[i]);
            perm (s, l+1, h);
            swap (s[l], s[i]);
        }
    }
}
```