① → Reversing a string -

```
String reverse(String s) {
    StringBuilder sB = new StringBuilder();
    for (int i = s.length()-1; i >= 0;
                                i--) {
        sB.append(s.charAt(i));
    }
    return sB.toString();
}
```

T → O(N)
S → O(N)

② → Special number -

Given a number N, find out whether it is divisible by 3.

N = "123"

```
boolean isDivisibleBy3(String N) {
    int sum = 0;
    for (int i = 0; i < N.length(); i++) {
        sum = sum + (N.charAt(i) - '0');
    }
```

```
return sum % 3 == 0;
}
```

T = O(N)
S = O(1)

③ flames –

s1 = saumya
s2 = ansh

remove all the letters that are common to both the strings from both the strings.

```
String flameGame(String s1, String s2) {
    int[] counts1 = new int[256];
    int[] counts2 = new int[256];
    for(int i=0; i< s1.length(); i+=1) {
        counts1[s1.charAt(i)] += 1;
    }
    for(int i=0; i< s2.length(); i+=1) {
        counts2[s2.charAt(i)] += 1;
    }
    int sum = 0;
    for(int i=0; i< 256; i+=1) {
        sum += Math.abs(counts2[i] - counts1[i]);
    }
```

~~s→1~~ ~~a→1~~
1 ~~q→2~~ ~~n→1~~
1 ~~y→1~~ ~~d→1~~
4 ~~m→1~~ ~~h→1~~
~~y→1~~

```
        return getRelationShip(sum);
    }

    String getRelationShip(int num) {
        num = num % 6;

        switch(num) {
            case 0:
                return "Siblings";
            case 1:
                return "Friends";
            case 2:
                return "Love";
            case 3:
                return "Affections";
            case 4:
                return "Marriage";
            case 5:
                return "Enemy";
            default:
                return "NOT FOUND";
        }
    }
```

T = O(max(M,N))
S = O(1)

④ Compress String -

→ AAACCCBBD
A3C3B2D1

```
String compress ( String s ) {

    if ( s.length() < 2 ) {
        return s;
    }

    StringBuilder sB = new StringBuilder();
    int count = 1;
    int i = 0;

    while ( i < s.length() ) {
        if ( i < s.length()-1 &&
             s.charAt(i) == s.charAt(i+1) ) {
            count += 1;
        } else {
            sB.append ( s.charAt(i) );

            if ( count > 1 ) {
                sB.append (count);
            }
            count = 1;
        }
        i += 1;
    }

    return sB.toString();
}
```

T = O(N)
S = O(N)

⑤→ Infinity stone-

s = "2345"
r = "5432"

2→5	3
3→4	1
4→3	1
5→2	3
	9

```
int rotations (String s, string r) {
    int count = 0;
    for (int i = 0; i < s.length(); i+=1 ){
        count += Math.abs(s.charAt(i) -
            r.charAt(i));
    }
    return count;
}
```

T = O(N)
S = O(N)

⑥ → Make all the characters in the string equal-

S → consists of only 'a' and 'b'

→ abaa

aaaa

```
int minOperations( String s) {

    int countA = 0,
        countB = 0;

    for (int i = 0; i < s.length(); i+=1){

        if ( s.charAt(i) == 'a') {
            countA += 1;
        } else {
            countB += 1;
        }
    }

    return Math.min( countA, countB);

}
```

T = O(N)

S = O(1)

① Palindrome –

s = c b a b c c

- at least there should be a substring of length 2 or more which is palindrome

```
boolean isPalindrome(String s) {
    for(int i=0; i<s.length(); i+=1) {
        if ( checkAroundCurrChar(s, i-1, i+1)){
            return true;
        }
        if ( checkAroundCurrChar(s, i, i+1)) {
            return true;
        }
    }
    return false;
}

boolean checkAroundCurrChar(String s, int left,
                            int right) {
    if( left>=0 && right < s.length() &&
        s.charAt(left) == s.charAt(right)){
        return true;
    }
    return false;
}
```

T = O(N)
S = O(1)

⑧→ Longest Palindrome length

```
String longestPalindrome(String s) {
    int maxLen = Integer.MIN_VALUE;

    for(int i=0; i< s.length(); i+=1) {
        int len1 = expandAround(s, i, i);
        int len2 = expandAround(s, i, i+1);
        int len = Math.max(len1, len2);
        if (len > maxLen ) {
            maxLen = len;
        }
    }
    return maxLen;
}

int expandAroundCenter(String s, int left,
                       int right) {
    while(left >=0 && right < s.length()
          && s.charAt(left) == s.charAt
                               (right)) {
        left -= 1;
        right += 1;
    }
    return right - left - 1;
}
```

a b b a
-1 0 1 2 3 4

$T = O(N^2)$

$S = O(1)$