String

→ Sequence of characters
OR
Array of characters

Ex:-

" Hello World"
" Welcome To Scaler ! "

Note :- Represented using double quote " "

Character

→ A symbol that represents a letter , number , special symbol etc.

→ ex:- 'A' , '$' , '-' , '9'

How do computers store characters if all they understand is Binary ?

→ Every character is mapped to a corresponding ASCII value.

'A' → 65	'a' → 97	'0' → 48
'B' → 66	'b' → 98	'1' → 49
⋮	⋮	⋮
'Z' → 90	'z' → 122	'9' → 57

→ Every Special character also has its corresponding ASCII value.

→ ASCII value of space ' ' is 32.

Some Operations:

①
```
char ch = (char) 65;
print (ch);
```

A

②
```
char ch = (char) ('a' + 1);
print (ch);
```

b

③
```
int x = 'a';
    print(x);
```
97

Problem 1

Given a string consisting of only alphabets (either lowercase or uppercase) Print the string by reversing its each characters (L·C → U·C
U·C → LC)

Ex:- "Hello"
Output :- "hELLO"

Quiz 1

Ex:- "aDgbHJe"
Output :- "AdGBhjE"

Observations

1. To convert small alphabet to capital alphabet, subtract 32

2. To convert capital alphabet to small alphabet, add 32.

Code

```
String toggle (char s[]) {
    for ( int i = 0 ; i < N ; i++ ){
        if ( s[i] >= 65 && s[i] <= 90){
            s[i] = s[i] + 32;
        } else {
            s[i] = s[i] - 32;
        }
    }
```

T.C → O(N)

S.C → O(1).

Substring

- → Similar to subarray.
- → Continuous sequence of characters within a string.

A single character can also be a substring. Full string can also be a substring.

ex :- $\text{"}\overset{0}{a}\ \overset{1}{b}\ \overset{2}{c}\text{"}$

a   b   c

a b   b c

a b c

Quiz 2

$\text{"}\overset{0}{b}\ \overset{1}{x}\ \overset{2}{c}\ \overset{3}{d}\text{"}$

b   x   c   d

b x   x c   c d

b x c   x c d

b x c d

$$\frac{N(N+1)}{2} = \frac{\overset{2}{\cancel{4}} \times 5}{\cancel{2}} = 10$$

Problem 2:- Given a substring S. Check if it is palindrome or not.

ex :- "NAYAN", "MADAM"

MALAYALAM

Ex!- S = "a n a m a d a m s p e"

0	1	2	3	4	5	6	7	8	9	10
a	n	a	m	a	d	a	m	s	p	e
			i				j			

start = 3
end = 7

true.

Approach:-

0	1	2	3	4
N	A	Y	A	N
		i		
		j		

i++;
j--

while(i< j)

A N N A

i
j

N A V E E N

i ≠ j

Code.

```
boolean    isPalindrome (char s[], int s, int e) {
    while ( s < e ) {
        if ( s[s] != s[e]) {
              return false;
        }
        else {
            s++;
            e--;
        }
    }
return    true;
```

T.C → O(N)

S.C → O(1)

Problem 3 Given a String S, calculate the length of longest palindromic substring in S.

Ex:- "a n a m a d a m m"

0	1	2	3	4	5	6	7	8
a	n	a	m	a	d	a	m	m

ans = 5

Quiz 3

"f e a c a b a c a b g f"

0	1	2	3	4	5	6	7	8	9	10	11
f	e	a	c	a	b	a	c	a	b	g	f

ans = 7.

Quiz 4

"a d a e b c d f d c b e t g g t e"

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	d	a	e	b	c	d	f	d	c	b	e	t	g	g	t	e

length = 9

Brute Force

Check for all possible substrings.

Code

```
int      longestPalindromeSS ( char s[] ) {

          int    N =  s.size();
          int    ans  = 0;
      // Fixing Starting Index
  for ( i = 0 ; i < N ; i++ ) {
          // Fixing Ending Index
          for ( int j = i ; j < N ; j++ ) {
                    if ( isPalindrome ( s , i , j ) ) {
                        ans = max( ans , j-i+1);
                    }
          }
  }
              return  ans;
```

T.C → O(N^3)

S.C → O(1)

Optimised Solution

len = 0

0 1 2 3 4 5 6 7 8 9 10 11

" f e a c a b a c a b g f "

1. Odd length

i

N A Y A N

2. Even length

l r

c

c1 c2

0 1 2 3 4 5 6 7 8 9 10 11 12

" f e a c a b a a b a c g f

l r

l

r

(l, r) $r - l + 1 - 2$

Code:

```
int       longestPalindromeSS( char s[]) {

        int   maxLength = 0;
        int   N = s.size();

for (c = 0 ; c < N ; c++) {

        // For odd length S.S.

    int   left = c   ;  right = c
    while ( left >= 0  &&  right < N){
        if ( s[left] != s[right] ) {
                break;
        }
        left--;
        right++;
    }

    maxLength = max(maxLength
                        right - left - 1);
   // For even length S.S.

   int   left = c   ;  right = c + 1;
  while ( left >= 0  &&  right < N){
      if ( s[left] != s[right] ) {
              break;
      }
      left--;
      right++;
  }
```

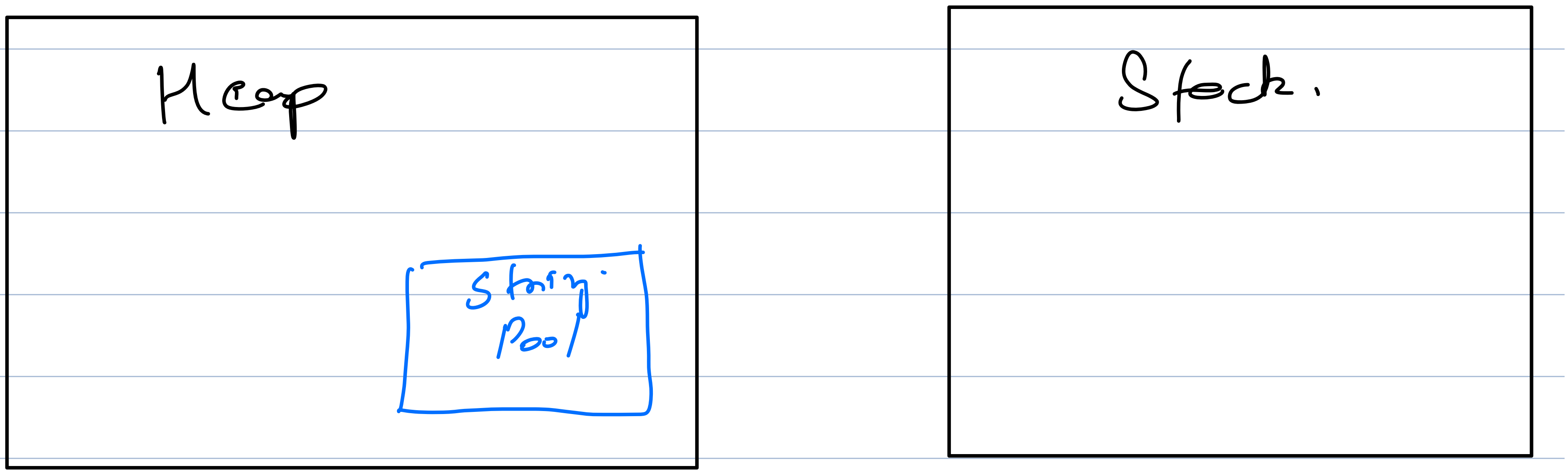
```
    maxLength = max(maxLength
                    right - left - 1);
}

    return maxLength;
```

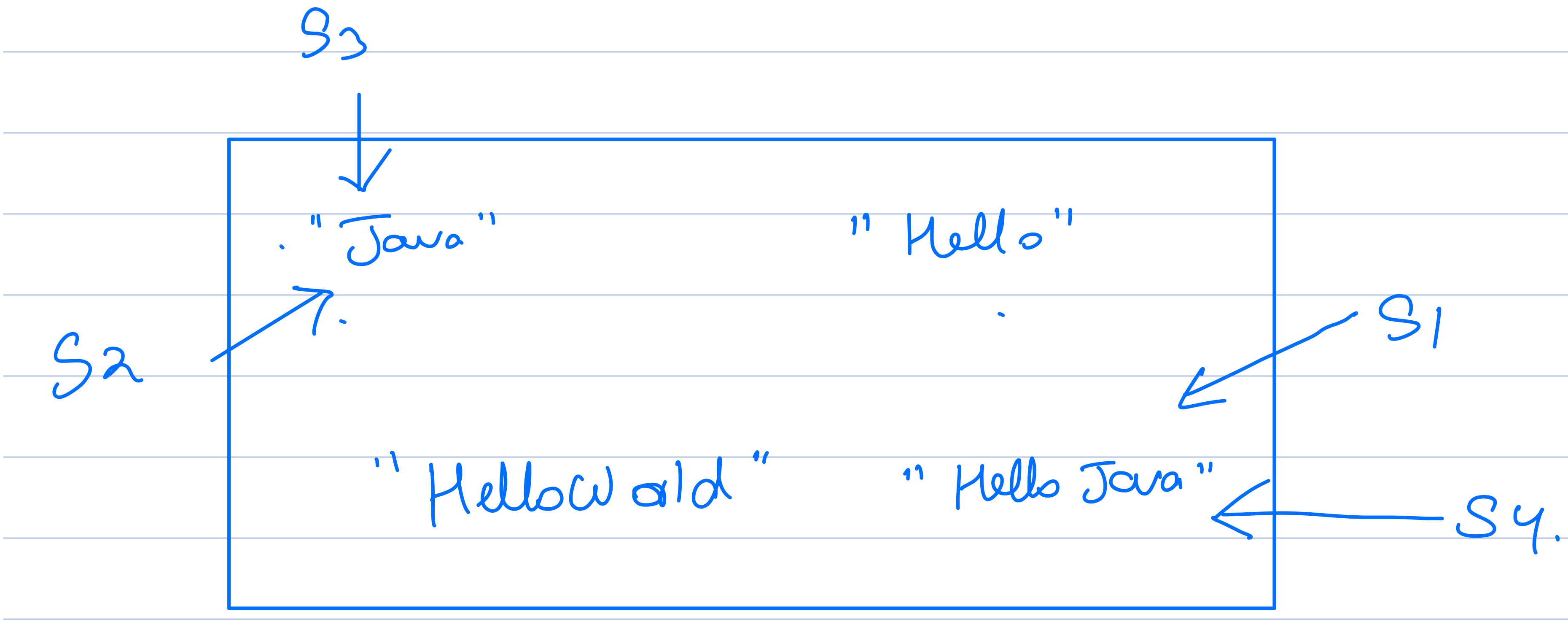
T.C → $O(N^2)$

S.C → $O(1)$

Immutability of Strings

In Java, C#, JavaScript, Python & Go, Strings are Immutable which means, its value can't be changed.





String Pool

① s1 = "Java" ③ String s3 = s1;

② s2 = "Java" ⑤ s1.concat("World")

④ s1 = "Hello";

⑥ String S4 = S1.concat(S3);

⑦ S1 = S4,

char '1'

int 1