Strings

Defination

↳ A String can be defined as a sequence of characters or in other words we can say it is an array of characters.

Ex:- " Welcome to Scaler "

" Hello World "

Q what are characters?

↳ Any letter, number or Symbol. It is represented by single quotes.

Ex:- 'A' , 'a' , '1' , ';'

★ Each character has corresponding decimal value associated with it known as ASCII value.

Ex:- 'A' to 'Z' → 65 to 90

'a' to 'z' → 97 to 122

'0' to '9' → 48 to 57

And, '?', '!' & '*'... also have ASCII values.

New Learning [Java / Python]

↳ characters are actually stored as UNICODE

Reasoning → ASCII → Limited digs [8 bits] → 2^8

UNICODE → [16 bits] → 2^{16}

★ UNICODE store the same mapping as that of ASCII with some additional mapping. UNICODE was required when Java / Python was created

Question :- Predict the output

(I)
```
char ch = (char) 65;
print (ch);
```
↳ A

(II)
```
char ch = (char)('a'+1);
print (ch)
```
↳ b

(III)
```
int x = 'a';
print (x)
```
↳ 97

Question :- Given a String of only Alphabets. Print lowerCase for all upperCase & upperCase for all lowerCase.

Ex :- Hello
↓ ↓ ↓ ↓ ↓
output :- hELLO

Quiz ! :- what is the output for String = "aDgbHJe"?

I/P → aDgbHJe

Ans O/P → AdGBhjE

Observations :-

① The difference b/w lower case & upper case ASCII value is 32.

Eg :- 'a' − 'A' = 32
'b' − 'B' = 32
'z' − 'Z' = 32

PSEUDO CODE

```
for ( i=0; i < n; i++){
    if ( s[i] >= 'A' && s[i] <= 'Z'){
        Print (s[i] +32);
    }
    else {
        Print (s[i]-32);
    }
}
```

TC - O(N)
SC - O(1)

SUBSTRING

↳ A Substring is a contiguous sequence of character within a string. A Substring concept in a string is similar to Subarray concept in array.

A Substring can be :-

1. Continous part of string.
2. Full string can be substring.
3. A single character can be substring.

Ex :- "abc" Substrings → "a", "b", "c", "ab", "bc", "abc"

Total No of Substring of length N String = $\frac{N*(N+1)}{2}$

Quiz 2 :- How many total Substring will be there for the String "b x c d" ?

↳ N = 4

$\Rightarrow \frac{N \times (N+1)}{2} \Rightarrow \frac{4 \times (5)}{2} \Rightarrow \boxed{10}$

Question :- Check whether the given Substring of String 's' is palindrome or not.

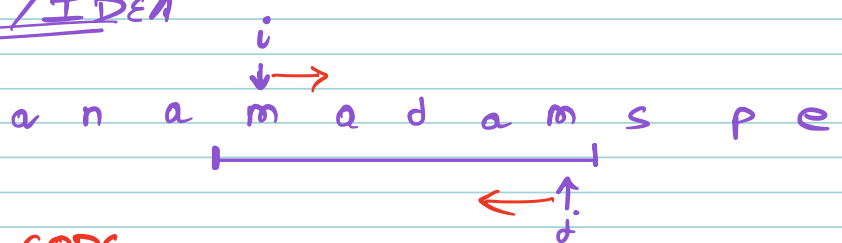
A palindrome is the sequence of characters that reads the same forward & backward
Ex:- "Nayan", "madam". etc.

Ex :- s = " a n a m a d a m s p e "
0 1 2 3 4 5 6 7 8 9 10

start = 3
End = 7

Ans ⇒ True.

Observation / IDEA



PSEUDO CODE

```
Function isPalindrome(char[] s, s, e){
    i = s, j = e
    while (i < j){
        if (s[i] != s[j]){
            return False;
        }
        i++
        j--
    }
```

return True;

}

TC → O(N)
SC → O(1)

Question :- Given a string 'S', find len of longest palindromic substring in 'S'

Ex :- a n a m a d a m

Ans = 5

Quiz 3 :- what is the length of longest Palindromic substring within string :-

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

Ans => 7

Quiz 4 :- what is the length of longest substring within string :-

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

Ans = 4

Brute Force

↳ Find all substrings & then check palindrome

$[i \quad j] \Rightarrow j - i + 1$

PSEUDO CODE

```
ans = 0;
for (i=0; i<n; i++){
    for (j=i; j<n; j++){
        if (isPalindrome(s, i, j)){
            ans = max(ans, j-i+1)
        }
    }
}
return ans;
```

start → `for (i=0; i<n; i++){`

End → `}`

TC	$O(N^3)$
SC	$O(1)$

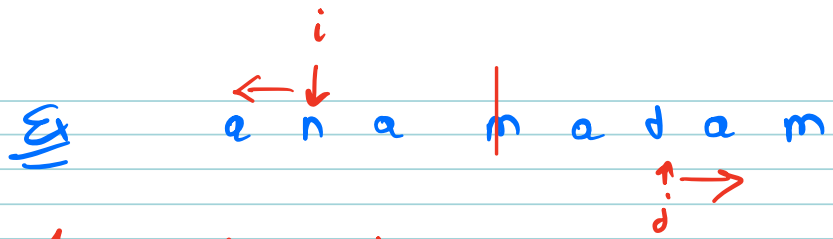
↳ OPTIMIZATION

Observations

① If smaller palindrome at centre is not palindrome then larger containing with same centre can never be palindrome smaller substring

IDEA 1 :- (I) max odd length palindromic substring

↳ odd have centre in palindromic string so let consider all the possible centre



Ex

Ans 1 3 1 .....

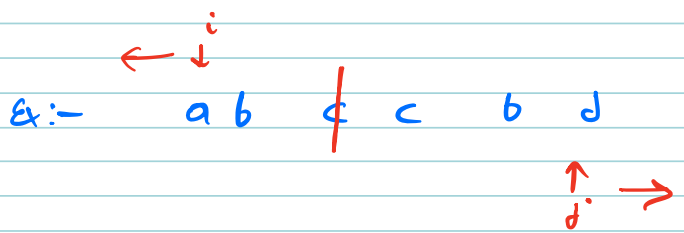
PSEUDO CODE

```
ans = 0
for (c = 0; c < n; c++){
    i = c;
    j = c;
    while (i >= 0 && j < n){
        if (s[i] != s[j]){
            break;
        }
        i--;
        j++;
    }
    ans = max(ans, j-1-i);
}
```

TC — $O(N^2)$

SC — $O(1)$

IDEA 2 :- max even length palindromic Substring



PSEUDO CODE

```
for (int c = 0; c < n; c++){
    i = c;
    j = c+1;
    while (i >= 0 && j < n){
        if (s[i] != s[j]){
            break;
        }
        i--;
        j++;
    }
    ans = max(ans, j-1-i);
}
```

TC - $O(N^2)$

SC - $O(1)$

IMMUTABILITY

[In Context of Programming]

↳ something which can't be changes.

* In Java / python & few other languages strings are immutable.

question :- So what happens when we append?

Ex :- Str = "Hello".

Str = Str + "World";

Print (Str)

↳ "Hello World"

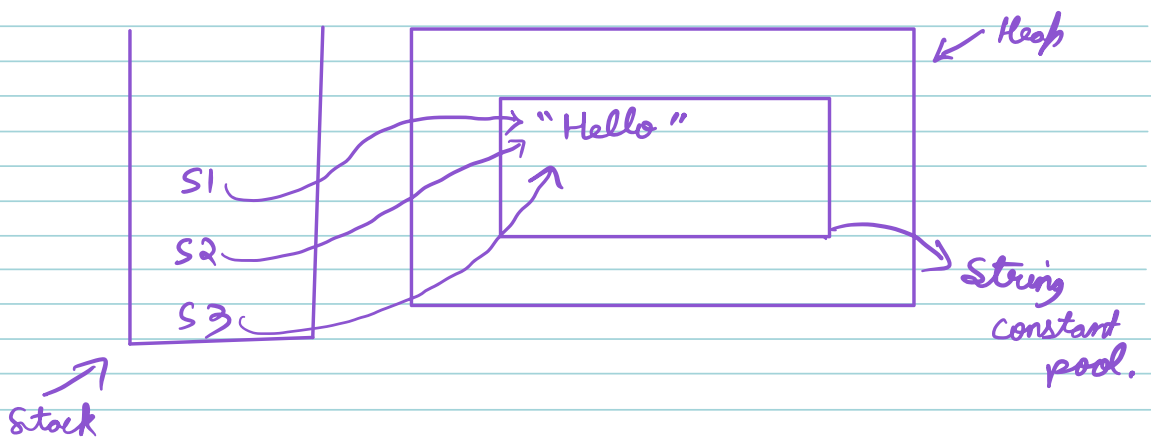
Ans :- A new string will be created & stored inside Str variable

↳ EXPLANATION

String S1 = "Hello"

String S2 = "Hello"

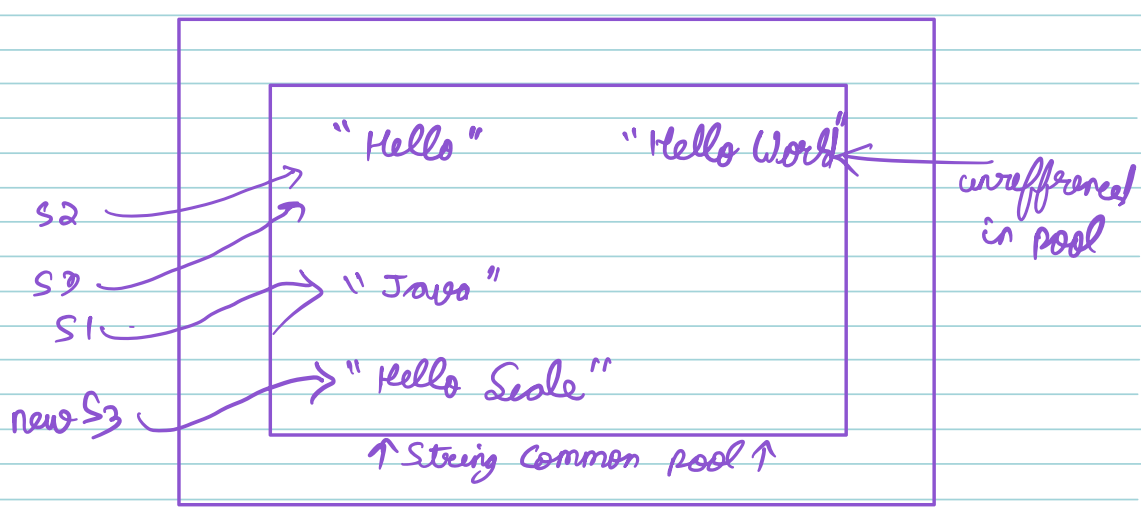
String S3 = S1;



★ If there is a string in "String Constant pool" then the variable will start pointing to it.

Example Extension

```
// changing the value of s1
→ s1 = "Java";
// Updating with concat() operation.
→ s2.concat("World");
// The concatenated String will be created as a new instance concatenated value.
→ String newS3 = s3.concat("Scaler");
→ sop(s1)      // Java
→ sop(s2)      // hello
→ sop(s3)      // hello
```



CONCLUSION :- whenever you are trying to change string a new string will be created

PSEUDO CODE

```
String str = "";
for (i: 0; i < n; i++){
    str += i;          // O(N)
}
```

O(N) →

TC → $O(N^2)$

Question :- Reverse the String.

CODE :-

```
String str = "Hello"
String reverse = "";
for( i = N-1; i >= 0; i--){
    reverse += str[i]; //O(1)
}
```

rev = "O"
= "OL"
= "OLL"
= "OLLE"
= "OLLEH"

TC → $O(N^2)$

↳ OPTIMIZATION [String Builder]

CODE :-
```
String Builder Str = new StringBuilder();
for (i = 0; i < n; i++){
    Str.append(i);      //O(1)
}
```

O(N) →
```
String newStr = Str.toString();
print(newStr);
```

TC → O(N)