String Basics

TABLE OF CONTENTS

String

→ order of characters matter.

Array of characters. ✓

Group of characters. ✗ abc $\neq$ bac

sequence of characters. ✓

Character

symbol that represents ->

1) letter / alphabets
2) digit
3) special symbol ('#', '÷', '@' etc.)

How to store characters?

Every character has a ASCII value mapped.

'A' → 65	'a' → 97	'0' → 48
'B' → 66	'b' → 98	'1' → 49
'C' → 67	'c' → 99	'2' → 50
.	.	.
.	.	.
.	.	.
'Y' → 89	'y' → 121	'8' → 56
'Z' → 90	'z' → 122	'9' → 57

1. char ch = 'a';

   print(ch); → a

2. int ch = 'a';

   print(ch); → 97

Arithmetic operator convert char to ASCII value.

3. int ch = 'a' + 1;

   print(ch);

   97+1 ↓ 98

4. char ch = (char) ('a'+1) (char)(98)

   print(ch) → b

Switch Case

< Question > : Given a string consisting of lower-case and upper-case alphabets.

Convert: (1) lowercase → uppercase

(2) uppercase → lowercase

1. "H e l l o" - "hELLO"

2. "a D g b H J e" - "AdGBhjE"

'a' → 97	'A' → 65	97-65=32
'b' → 98	'B' → 66	98-66=32
'c' → 99	'C' → 67	99-67=32
⋮	⋮	
'z' → 122	'Z' → 90	122-90=32

</> Code

```
N = str.length() ← size of your string.

for(i=0; i<N; i++)
{                                        // str.charAt(i)
    char ch = str[i];
    if(ch>='a' && ch<='z')
    {
        str[i] = (char)(ch-32)
    }
    else
    {
        str[i] = (char)(ch+32)
    }
}
```

TC = O(N)   SC = O(1)

In some programming language updating char in string is not allowed. (EG. Python, Java, etc)

=> Create another string.

</> Code

```
ans = " "
N = str.length()  <-- size of your string.

for(i=0; i< N ; i++)
{                                  // str.charAt(i)
  char ch = str[i];
  if(ch>='a' && ch <= 'z')
  {
     ans=ans+(char)(ch-32)
  }                           } appending the char
  else
  {
     ans = ans + (char) (ch+32)
  }
}
```

TC = O(N)   SC = O(1)

→ C++ ✓

→ Java TC = $O(N^2)$

appending char in string = O(length)

To Solve:

O(N) 1) Convert String to char array

O(N) 2) Solve using char array.

O(N) 3) Conver char array back to String.

→ new String(char array)

TC = $O(3N) = O(N)$

SC = $O(N)$

"Hello"

Substring

1. Contiguous part of a string. ✓
2. A single character is also a substring. ✓
3. Whole string is also a substring. ✓
4. Empty string (" ") is not a substring. ✗
5. String of length N. How many substrings will be there? ✓ $\frac{N*(N+1)}{2}$

abc

a
a b
a b c
b
b c
c

bxcd → $\frac{4*(4+1)}{2} = \frac{4*5}{2} = \frac{20}{2} = 10$

b
bx
bxc
bxcd

x
xc
xcd

c
cd

d

Check for substring if it's a palindrome or not.

→ str == reverse of str

```
                si                ei
str -  a , b , m , a , d , a , m , t , a , m
       0   1   2   3   4   5   6   7   8 . 9
```

si = 2 , ei = 6

Ans = True.

```
boolean isPalindrome( String str, int si, int ei ) {

            left = si
            right = ei
            while (left < right)
            {
                    if (str[left] ! str[right])
                    {       return false;
                    }
            } left ++;   right --;
               return true;
}
```

TC: O(N)

SC: O(1)

< Question > : Given a string s. Find the length of the longest palindrome substring in s.

s → "anmadamm" 1 ≤ N ≤ 10^3

Ans= 5

s → "feacabacabgf"

Ans = 7

s → "adaebcdfdcbetggte"

Ans=9

[BF Idea] - For each substring check if it is palindrome or not.

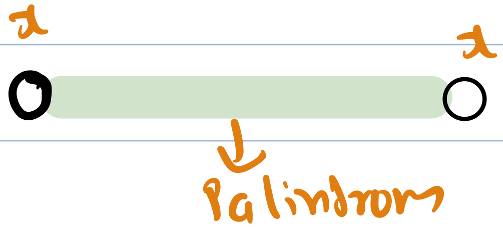
```
fun longestPalindrom(char[] chr)
{
        N= chr.length;
        ans=0;
```

TC: $O(n^3)$

SC: $O(1)$

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

Break : 10.27 PM



Sol^n: ~~for~~ Keep every char (1 char) | 2 char as middle element, expand the substring till it is palindrome.

Odd length → 1 middle
even length → 2 middle.

[Idea - 2] - " a , d , a , e , b , c , d , f , d , c , b , e , t , g , g , t , e "

Odd length:

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

odd ans = ~~0~~ ~~1~~ ~~3~~ ~~7~~ 9   len=1

Even length:

i j

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

Even Ans = ~~0~~ ~~2~~ ~~4~~ 6

len = ~~0~~ ~~2~~ ~~4~~ 6

Final Ans = Max(evenAns, oddAns)

= Max (6, 9)

= 9

evenAns = 4

" a b c c b c "

oddAns = 3

Ans = 4

</> Code

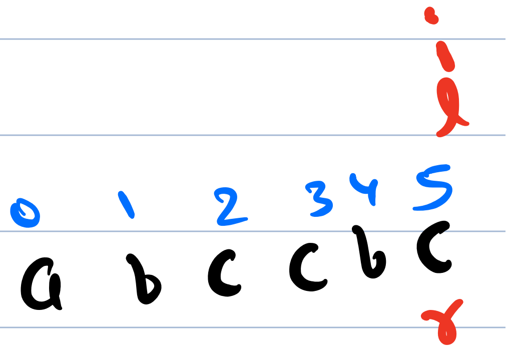
len= 1

OddAns = ~~0~~ ~~1~~ 3

```
for (i=0; i<n ; i++)
{
    l = r = i
    len = 1

    while( l>=0 && r<n && s[l]==s[r])
    {
        if (len > oddAns)
        {   oddAns = len;
        }
            l--; r++;

        len= len+2;
    }
}
```



</> Code

len= 2

evenAns = ~~0~~ ~~2~~ 4

```
for (i=1; i<n; i++)
{
        l = i-1
        r = i
        len = 2
while( l>=0 && r<n && s[l]==s[r])
{
        if (len > evenAns)
        {   evenAns= len;
        }
            l--; r++;

          len= len+2;
}}
```

```
                                  l  i
0  1  2  3 4 5
a  b  c  c b c
                                     r
```

finalAns= Max(evenAns,oddAns);

return finalAns;

TC= $O(n^2)$ SC= $O(1)$

(Java, Python, C#, golang etc.)

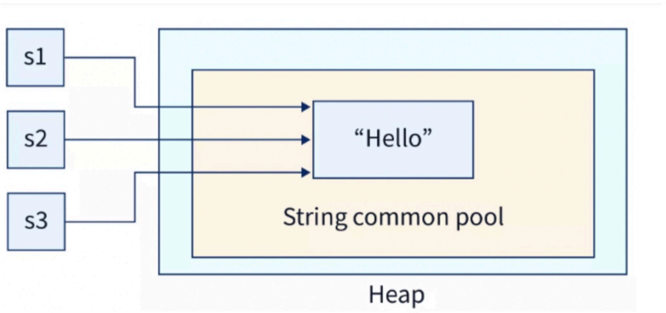
Immutability of the strings

→ value can't change

String s1 → "Hello"

String s2 → "Hello"

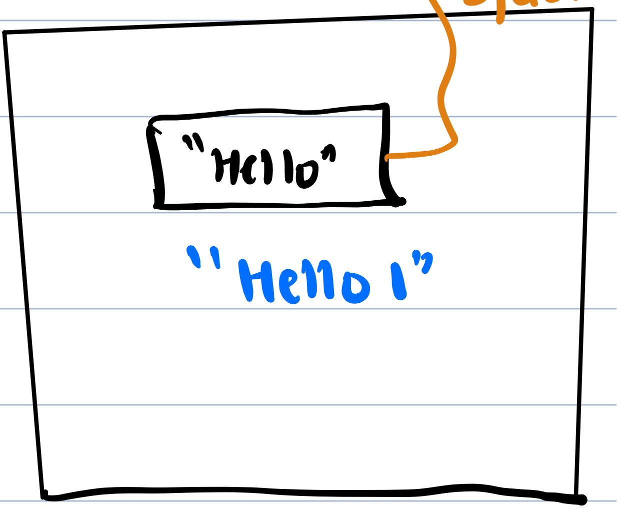
String s3 = s1



Garbage collected to free up the space.

String a = "Hello"

a = a + '1'



// new string is created

TC = O(length)

How to update efficiently:

1) Convert String to array.
2) Do your operation
3) Convert back to string.

Char[] chr = ['a','b','c']

↙

new String(chr)

*) Mutabale form in some languge.

Eg. Stringbuilder in java.

③ ↶

String $\xrightarrow[①]{}$ Stringbuilder (complete steps)

Sb.append('a'); ← TC = O(1)

Doubt session

a	b	c	d	e	f	g	h	← chr

```
String ans = " ";

for (i=0 ; i<N; i++)
{
    ans = ans + chr[i];
}
```

i=0	" "	→	"a"	1	~0
i=1	"a" + 'b'	→	"ab"	2	~1
i=2	"ab" + 'c'	→	"abc"	3	~2
i=3	"abc" + 'd'	→	"abcd"	4	~3
⋮					
	N-1 +				
i=N	" " + ''	→	" "	N	~N-1

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____

____