String $\Rightarrow$ Sequence of characters.

Eg : 1) " Hello World"

2) " Welcome To Scaler"

$$\begin{tikzcd} & \text{String is represented by (")} \arrow[dl] \arrow[dr] & \\ \text{Character Array} & & \text{String} \end{tikzcd}$$

<u>H.W.</u> : find the difference b/w a Character Array & a String in your language.

Character

Integer $\Rightarrow$ 8 Bytes = 32 bits

Char $\Rightarrow$ 1 Byte = 8 bits

$$\underline{2} \times \underline{2} \times \underline{2} \times \underline{2} \times \underline{2} \times \underline{2} \times \underline{2} \times \underline{2} = 2^8$$

$$2 \to 0, \quad 2 \to 1$$

$$2^8 = 256$$

$$\Downarrow$$

$$[0, 255]$$

$$\begin{tikzcd}
A \arrow[r] & 65 \arrow[r, dash, "32"] & a \arrow[r] & 97 \\
B \arrow[r] & 66 \arrow[r, dash, "32"] & b \arrow[r] & 98 \\
C \arrow[r] & 67 \arrow[r, dash, "32"] & c \arrow[r] & 99 \\
D \arrow[r] & 68 & d \arrow[r] & 100 \\
\vdots & \vdots & \vdots & \\
Z \arrow[r] & 90 \arrow[r, dash, "32"] \arrow[r, bend left=60, "+32"] & z \arrow[r] \arrow[l, bend left=60, "-32"] & 122
\end{tikzcd}$$

[65, 90] [97, 122]

Operations

char ch = 'A';

char ch = $\underbrace{\text{(char)}}_{\Downarrow}$ 65;

Type-Casting. (H.W. Syntax in your language)

char ch = 'a';

char ch = $\underbrace{\text{(char)('a' + 1)}}_{\text{'b'}}$;

```
int x = 'a'
print (x); → 97
```

Typecasting ⇒ Bigger ⇒ Smaller

Q Given a char array of only alphabets (LowerCase & Upper Case)

Print all the characters of the string s.t.
1) You print upper as lower
2) You print lower as upper.

Eg : S : "Hello"
O/p : "hELLO"

Q aDgbHJe

O/p : AdGBhjE

Soln Code

```
Function toggle (char a[ ]) {
    int n = a.size();
    for (i = 0; i < n; i++) {
        if (a[i] >= 65 && a[i] <= 90) {
            print((char)(a[i] + 32));
        }
        else {
            print((char)(a[i] - 32));
        }
    }
}
```

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

Substring

Continuous sequence of characters within a string

Array $\Longrightarrow$ Subarray

String $\Longrightarrow$ Substring.

1) A single char also a substring

2) Entire string also a substring.

Q ==

# substrings for "bxcd"

$$\frac{(n)(n+1)}{2} = \frac{4 \times 5}{2} = 10$$

b	x	c	d
bx	xc	cd	
bxc	xcd		
bxcd			

Q == Given a String of size N & 2 integers l, r representing the start & end of a substring of S.

Check whether the above mentioned substring is a pallindrome or not.

Eg: S = "$\overset{0}{a}\,\overset{1}{n}\,\overset{2}{a}\,[\underline{\overset{3}{m}\,\overset{4}{a}\,\overset{5}{d}\,\overset{6}{a}\,\overset{7}{m}}]\,\overset{8}{s}\,\overset{9}{p}\,\overset{10}{e}$"

$l = 3$

$r = 7$

True.

Sol^n

Code

```
function ispallindrome (S[], start, end) {
    while ( start < end) {
        if (S[start] != S[end]) {
                return false;
        }
        start ++;
        end --
    }
    return true;
}
```

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

Q. Given a string S.
find the length of longest pallindromic substring of S.

Eg S = " ana madamm "
⇓
5

Quiz

f e $\underline{\text{a c a b a c a}}$ b g f

$\Downarrow$

7

Quiz

" a d a $\underline{\text{e b c d f d c b e}}$ t g g t e "

$\Downarrow$

9

Sol^n 1) Brute force

$\forall$ substring $\Rightarrow$ Check if it is a pallindrome & compare lengths to find max.

Code

```
int longestPallindrome (char S[]) {
    int ans = 1;
    for (i=0; i < S.size(); i++) {
            for (j= i ; j < S.size(); j++) {
                    if ( isPallindrome (S, i, j)) {
```

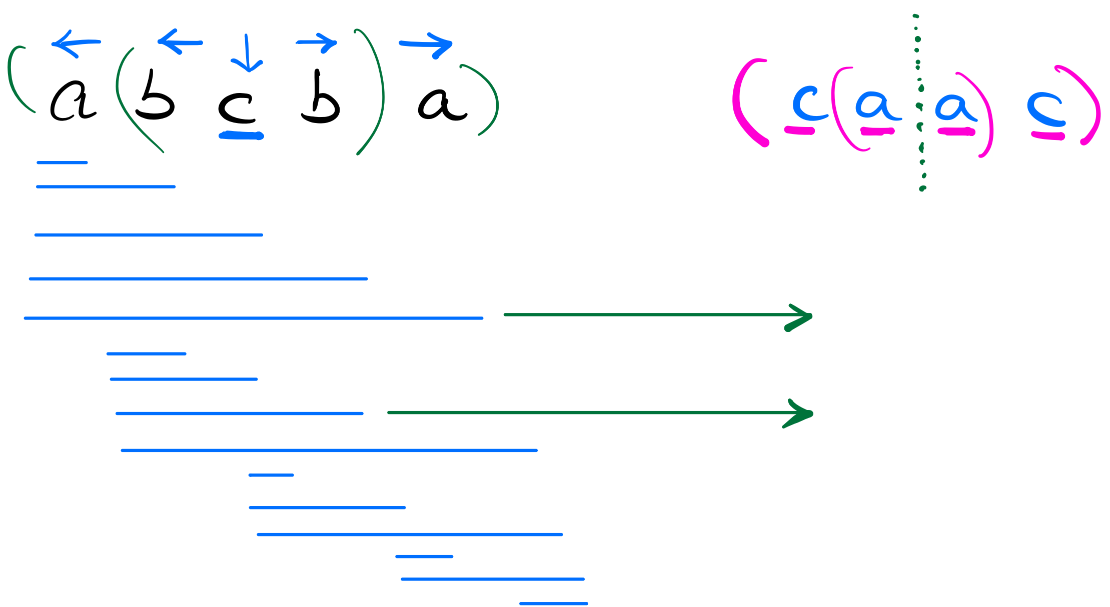
```
                length = j-i+1;
                ans = max(ans, length);
            }
        }
    }
    return ans;
}
```

$$T.C. = O(\underset{\substack{\downarrow \\ \forall\ \text{substring}}}{N^2} \times \underset{\substack{\downarrow \\ \text{Check} \\ \text{pallindrome}}}{N})$$

$$T.C. = O(N^3)$$

$$S.C. = O(1)$$

2) Optimise



Odd Substring

$\Rightarrow \forall i \rightarrow$ Consider A[i] as centre element of pallindromic substring & expand on both sides to find longest pallindrom substring with A[i] as the centre element.

Code

```
ans = 1;

for ( c = 0;  c < N;  c++) {   // O(N)
        length = 1;
        left = C - 1;
        right = C + 1;
O(N) // while ( left >= 0 && right < N) {

            if ( S[left] == S[right] ) {
                    length = length + 2;
            }
            else {
                    break;
            }
            left --;
            right ++;
        }
        ans = max (ans, length);
}
```

Even Substring

```
       left : right
    c  b  a : a  b  c
```

```
for (i=0, j=1; j<N;  i++; j++) { //N
        length = 0;
        left  =  i;
        right =  j;
N // while ( left >= 0 && right <N) {
            if ( S[left] == S[right] ) {
                    length = length + 2;
            }
            else {
                    break;
            }
            left --;
            right ++;
     }
```

```
    ans = max(ans, length);
}
```

T.C. = $O(N^2)$

S.C. = $O(1)$

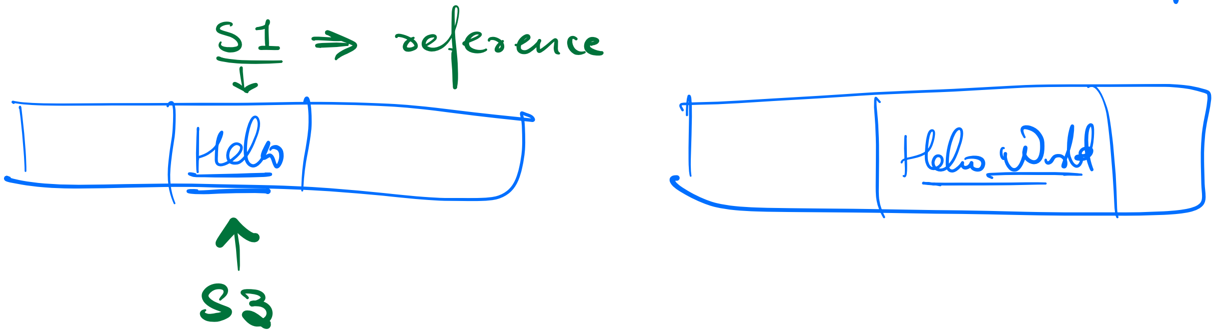
Java, C#, Python, JS, Go

Strings are immutable.

⇓

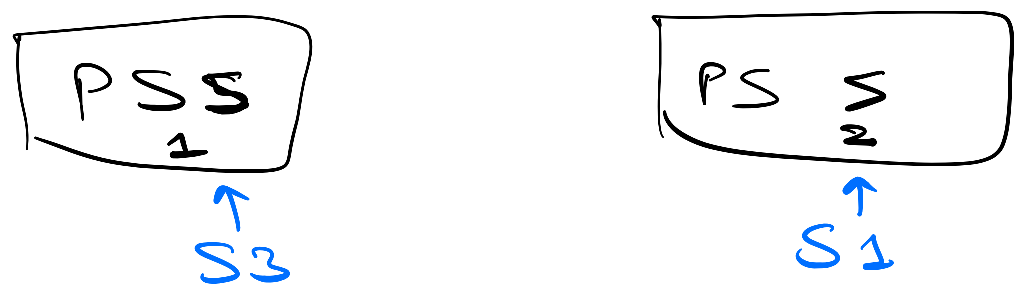
Value cannot be changed

```
String S1 = "Hello"; // String literal
String S2 = "Hello"; // String literal
String S3 = S1; // same reference
```



```
S1 = S1.concat("_World");

print(S3);   // (Hello)
```



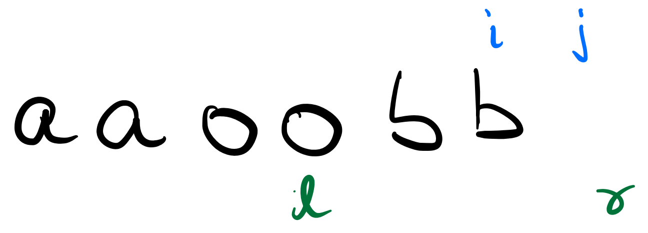
Hashing Basics

HashMap & HashSet

$\Downarrow$

Doubt

ans = 2



length = 2

Intermediate

Contest $\Rightarrow$

AP : $\underline{2}$, $\underline{4}$, 6, 8, 10, 12, 14

↑ sort

I/p : { 6, 10, 14, 4, 2, 8, 12 }

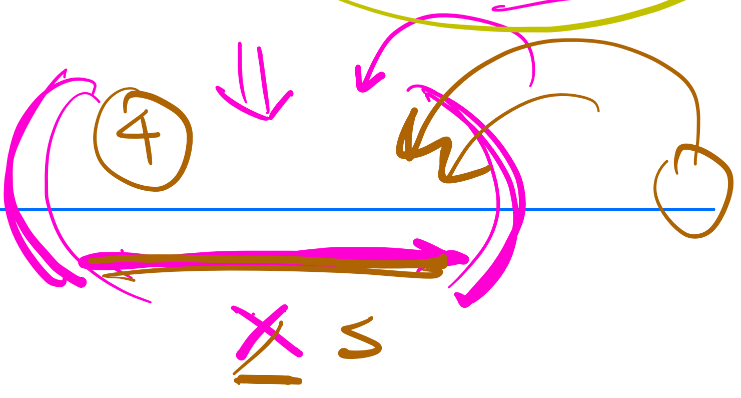
diff = $\underline{A[1] - A[0]}$

TA ↙

X = 5

elements ≤ B

Hint



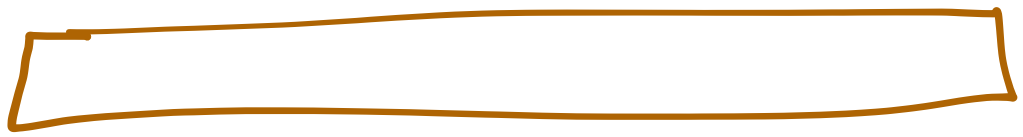
≤ B

X = 5

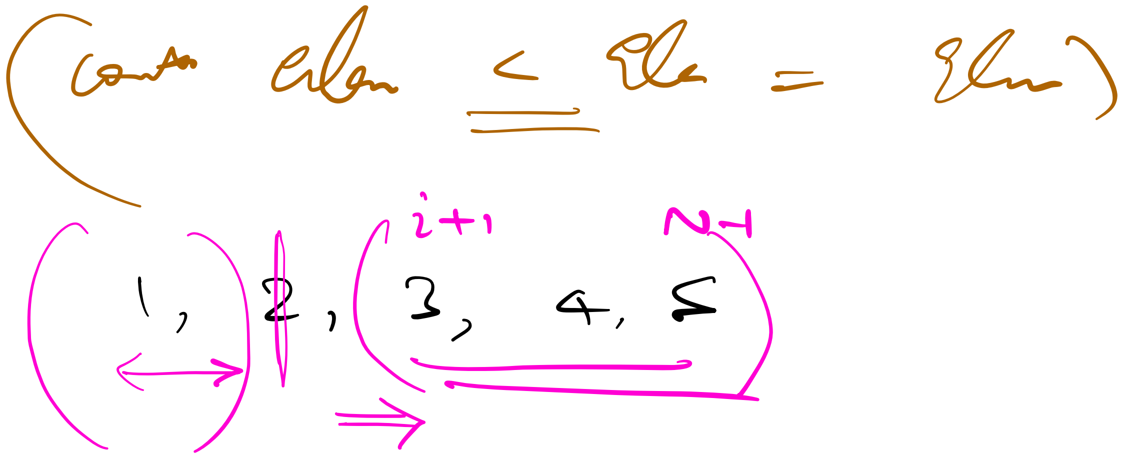
B = 4

$[(6, 2, 5, 3, 1,) 4, 7, 9, 4]$

Count = # elements > B



(count elem ≤ elem = elem)



$[0, i-1]$

$\|$

(i)

$(N-1) - (i+1) + 1$

$N-1-i-\cancel{1}+\cancel{1}$

$(N-i-1)$