Avg Batch PSP → 74% → >80% by next Monday

- Strings
- Hashing
- Bit Manipulation

Strings

1) array of char ✓
2) group of char ✗
3) collection of char ✗
4) sequence of char ✓

ordered arrangement of characters.

(a b c) abc / bac > different

"eat" ≠ "tea"

Java or similar languages → "string" 'character' 'a' 'b'

ASCII

'A' — 'Z' → 65 — 90
'a' — 'z' → 97 — 122
'0' — '9' → 48 — 57

} ✓

```
char ch = (char) 66;        → 'B'
char ch = (char) ('a' + 2); 'c'
                  ↓
                  97
```

```
int x = 'a' + 'A';   → 162
        ↓     ↓
        97    65
```

Q → Given a string consisting of alphabets, print all the characters after toggle. small → upper, upper → small

Eg → "Help" "PLAy"
O/p → "hELP" "plaY"

"0 1 2 3 4 5 6"
"aDgb HJe" → "AdGB hjE"

char → int

```
for i → 0 to (N-1) {   // N = str.length()
    ch = s[i]          // s.charAt(i)
    if ('A' <= ch && ch <= 'Z') {   // uppercase
        ch = (char)(ch + 32)        // 97 - 65 = 32
    } else {                        // 'a' - 'A' = 32
        ch = (char)(ch - 32)        // 'a' > 'A'
    }
    print(ch)
}
```

if ('a' > 'A')
print ("yes") ✓

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

|'A' - 'a'| = 32

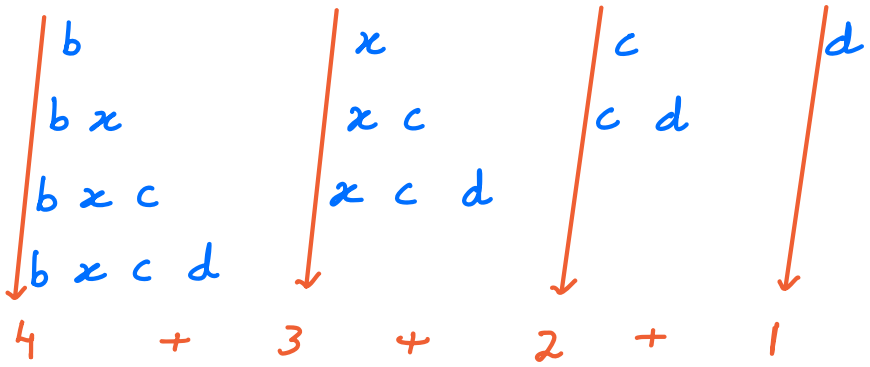
string → array of characters | print (int) 'a' → 97

Substring → subarray of characters
continuous part of string

"abc"
"ab" → substring
"ac" → not a substring (subset)

b x c d

Ans = 10



substrings = N + (N-1) + (N-2) + ... + 2 + 1 = $\frac{N*(N+1)}{2}$

Q → check if the given string is a palindrome.

Eg → "mom" → Ans = true

"madam" → Ans = true

"utkarsh" → Ans = false

check → str == reverse(string) ✓ SC = O(N)

" 0 1 2 3 4 5 6 "

racecar

i = 0 j = N-1 // substring [L R] → i = L j = R

```
while (i <= j) {
    if (s[i] != s[j]) return false
    i++
    j--
}
return true
```

TC = O(N)

SC = O(1)

10:20 PM

Q → Given a string, find the length of longest odd length palindromic substring.

Eg → "a b c b c b x b" Ans = 5

"f e a c a b a c s b g f" Ans = 7

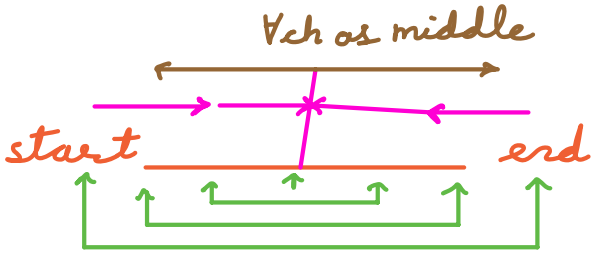
"a d a e b c d f d c b e t g g t e" Ans = 9

Bruteforce → ∀ odd length substrings → check palindrome.

```
ans = 0
for i → 0 to (N-1) {  // start
    for j → i to (N-1) {  // end      i — j      > N^2
        len = j - i + 1
        if ( len % 2 == 1 && isPalindrome(str, i, j))
                                          TC = O(N)
            ans = max(ans, len)
    }
}
```

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

"a d a e b c d f d c b e t" ans = t & 9



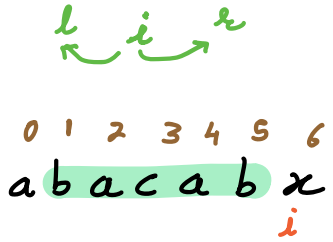
Sol → ∀ i as center → O(N)

find longest odd len palindrome.

O(N)

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

```
ans = 0
for i → 0 to (N-1) {  // middle
    len = 1
    l = i-1      r = i+1
    while (l >= 0 && r < N) {
        if (s[l] != s[r])
            break
        l--      r++
        len += 2
    }
    ans = max(ans, len)
}
return ans
```



l = 2 + 0
r = 4 & 6
len = 1 2 5
ans = 0 + 2 5 ✓

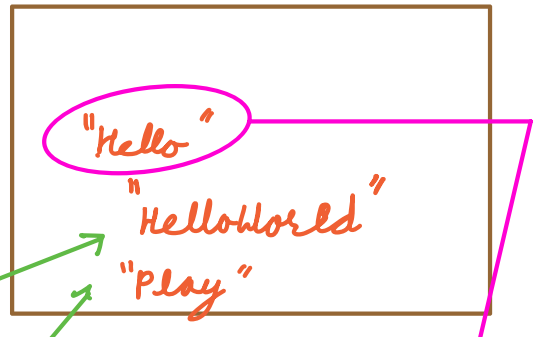
TC = $O(N^2)$ SC = $O(1)$

H.W → Even length palindrome
Similar code as above with → l = i

Immutability (finite meaningful english words) ✓

Java, C#, JS, Python etc.

```
String a = "Hello";
String b = "Hello";
```



a = a + "World"
// HelloWorld

b = "Play"

unused strings are removed from (Part of Heap Memory) memory via

garbage collecter.

Any change → new string
is string pool.

String a = ""
a = a + 'x' // x
a = a + 'y' // xy
a = a + 'z' // xyz

concatenation → TC = O(length of string)

Java → Strings (immutable)
→ StringBuilder (mutable)
