

# Solution 15: Palindrome Check with Two Pointers

---

## 1) Which strings are palindromes?

---

- "level" : True
  - "robot" : False
  - "ABBA" : True
  - "" : True
  - "A" : True
- 

## 2) Fill in the blanks: the two pointers idea

---

- `left` starts at 0
- `right` starts at `len(s) - 1`

We keep looping while `left < right`.

Inside the loop:

- If `s[left] != s[right]`, then `palindrome = False`.
  - If they match, we do:
    - `left = left + 1`
    - `right = right - 1`
- 

## 3) Trace the output: "ABCBA"

---

It prints two lines (then `left == right` and the loop stops):

```
0 4 A A  
1 3 B B
```

---

## 4) Trace the output: "ABCD~~B~~A"

---

It prints three lines (then it finds a mismatch at C vs D and breaks):

```
0 5 A A  
1 4 B B  
2 3 C D
```

---

## 5) Empty string

---

`len(s) - 1` is `-1`, so `left < right` is `0 < -1` (False). The loop never runs.

Output:

```
True
```

---

## 6) Debugging (2 bugs)

---

Wrong lines: - Line A: `right = len(s)` - Line B: `right = right + 1`

Corrected lines: - Line A: `right = len(s) - 1` - Line B: `right = right - 1`

---

## 7) Fill in the missing code

---

One correct completion:

```

s = "racecar"

palindrome = True
left = 0
right = len(s) - 1

while left < right:
    if s[left] != s[right]:
        palindrome = False
        break
    left = left + 1
    right = right - 1

print(palindrome)

```

---

## 8) Print the first mismatch (if any)

---

One possible solution:

```

s = "ABCDXDCBA"

left = 0
right = len(s) - 1
found = False

while left < right:
    if s[left] != s[right]:
        print(s[left], s[right])
        found = True
        break
    left = left + 1
    right = right - 1

if found == False:
    print("No mismatch")

```

Output:

C	D
---	---