

# Palindrome Number

## Easy

Given an integer  $x$ , return `true` if  $x$  is a **palindrome**, and `false` otherwise.

A **palindrome** is a number that reads the same forward and backward. For example, `121` is a palindrome while `123` is not.

---

## Examples

### Example 1:

Input:  $x = 121$

Output: `true`

Explanation: 121 reads as 121 from left to right and from right to left.

### Example 2:

Input:  $x = -121$

Output: `false`

Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

### Example 3:

Input:  $x = 10$

Output: `false`

Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

---

## Constraints

- $-2^{31} \leq x \leq 2^{31} - 1$
- 

## Follow-up

Could you solve it without converting the integer to a string?

---

## Solutions

## C Solution (Multiple Test Cases)

c

```
#include<stdio.h>
int main() {
    int a;
    scanf("%d",&a);
    while(a-->0) {
        long long n;
        scanf("%lld",&n);
        long long S=0;
        long long m=n;
        while(m>0) {
            S=S*10+m%10;
            m/=10;
        }
        if(S==n)
            printf("YES\n");
        else
            printf("NO\n");
    }
}
```

## C++ Solution

cpp

```

class Solution {
public:
    bool isPalindrome(int x) {
        // Negative numbers are not palindromes
        if (x < 0) return false;

        // Single digit numbers are palindromes
        if (x < 10) return true;

        // Reverse the number and compare
        long long original = x;
        long long reversed = 0;

        while (x > 0) {
            reversed = reversed * 10 + x % 10;
            x /= 10;
        }

        return original == reversed;
    }
};

```

## Python Solution

```

python

class Solution:
    def isPalindrome(self, x: int) -> bool:
        # Negative numbers are not palindromes
        if x < 0:
            return False

        # Convert to string and check if it reads the same forwards and backwards
        s = str(x)
        return s == s[::-1]

```

## Java Solution

```

java

```

```
class Solution {
    public boolean isPalindrome(int x) {
        // Negative numbers are not palindromes
        if (x < 0) return false;

        // Single digit numbers are palindromes
        if (x < 10) return true;

        // Reverse the number and compare
        int original = x;
        int reversed = 0;

        while (x > 0) {
            reversed = reversed * 10 + x % 10;
            x /= 10;
        }

        return original == reversed;
    }
}
```

---

## Complexity Analysis

**Time Complexity:**  $O(\log n)$  where  $n$  is the input number. We divide the number by 10 for every iteration.

**Space Complexity:**  $O(1)$  constant extra space used.

---

## Related Topics

- Math
  - Two Pointers
- 

## Similar Questions

- Reverse Integer (Medium)
- Valid Palindrome (Easy)
- Palindrome Linked List (Easy)