make\_palindrome Specification

# Prompt:

```python  
def make\_palindrome(string: str) -> str:  
 """   
 Find the shortest palindrome that begins with a supplied string.  
 Algorithm idea is simple:  
 - Find the longest postfix of supplied string that is a palindrome.  
 - Append to the end of the string reverse of a string prefix that comes before the palindromic suffix.  
 """  
```

# Step-by-Step Explanation and Specifications:

## Step 1: Find the Longest Palindromic Postfix

### Specification:

```python  
@icontract.require(  
 lambda string: isinstance(string, str) and string != "",  
 "The input must be a non-empty string"  
)  
@icontract.ensure(  
 lambda string, result: result == result[::-1],  
 "The result must be a palindrome"  
)  
@icontract.ensure(  
 lambda string, result: string.endswith(result),  
 "The result must be a suffix of the input string"  
)  
def find\_longest\_palindromic\_postfix(string: str) -> str:  
 pass  
```

### Code:

```python  
def find\_longest\_palindromic\_postfix(string: str) -> str:  
 for i in range(len(string)):  
 if string[i:] == string[i:][::-1]:  
 return string[i:]  
 return ""  
```

## Step 2: Append the Reverse of the Prefix

### Specification:

```python  
@icontract.require(  
 lambda string: isinstance(string, str) and string != "",  
 "The input must be a non-empty string"  
)  
@icontract.require(  
 lambda postfix: string.endswith(postfix),  
 "The postfix must be a suffix of the string"  
)  
@icontract.ensure(  
 lambda string, postfix, result: result.startswith(string),  
 "The result must start with the original string"  
)  
@icontract.ensure(  
 lambda string, postfix, result: result == result[::-1],  
 "The result must be a palindrome"  
)  
def create\_palindrome(string: str, postfix: str) -> str:  
 pass  
```

### Code:

```python  
def create\_palindrome(string: str, postfix: str) -> str:  
 prefix = string[:len(string) - len(postfix)]  
 return string + prefix[::-1]  
```

## Final Function: make\_palindrome

### Specification:

```python  
@icontract.require(  
 lambda string: isinstance(string, str) and string != "",  
 "The input must be a non-empty string"  
)  
@icontract.ensure(  
 lambda string, result: result.startswith(string),  
 "The result must start with the original string"  
)  
@icontract.ensure(  
 lambda string, result: result == result[::-1],  
 "The result must be a palindrome"  
)  
def make\_palindrome(string: str) -> str:  
 pass  
```

### Code:

```python  
def make\_palindrome(string: str) -> str:  
 postfix = find\_longest\_palindromic\_postfix(string)  
 return create\_palindrome(string, postfix)  
```

## Testing

### Test Cases:

```python  
def check(candidate):  
 assert candidate("race") == "racecar"  
 assert candidate("ab") == "aba"  
 assert candidate("a") == "a"  
 assert candidate("mad") == "madam"  
  
check(make\_palindrome)  
```