

```

nd_palindrome(input: String) -> bool {
  let input_byte: Vec<u8> = String::from(input).into_bytes();
  let input_byte_length: usize = input_byte.len();
  let mut pure_letter_byte_vector: Vec<u8> = Vec::new();
  for i: usize in 0..input_byte_length {
    let curr_char: u8 = input_byte[i];
    if curr_char >= 65 && curr_char <= 90 {
      pure_letter_byte_vector.push(curr_char+32);
    }
    else if (curr_char >= 97 && curr_char <= 122) || (curr_char >= 48 && curr_char <= 57){
      pure_letter_byte_vector.push(curr_char);
    }
  }
  let pure_letter_byte_vector_length: usize = pure_letter_byte_vector.len();
  let mut first_half: Vec<u8> = Vec::new();
  let mut other_half: Vec<u8> = Vec::new();
  let midpoint: usize = pure_letter_byte_vector_length / 2;
  if pure_letter_byte_vector_length % 2 == 1 {
    first_half = pure_letter_byte_vector[..midpoint].to_vec();
    other_half = pure_letter_byte_vector[midpoint..].to_vec();
  }
  else {
    first_half = pure_letter_byte_vector[..midpoint].to_vec();
    other_half = pure_letter_byte_vector[midpoint..].to_vec();
  }
  let half_length: usize = first_half.len();
  for i: usize in 0..half_length {
    if first_half[i] != other_half[half_length-(i+1)] {
      return false;
    }
  }
  return true;
}
find_palindrome

```

```

palindrome_verify_assert() {
  let palindrome_array: [&str; 30] = ["Racecar","War Thunders!","Peppa Pig","22/02/2022","Hornet's Tenor","Was it a car or a cat I saw?",
  Hello, World!","A man, a plan, a canal, Panama!","Hotdog Vaccume","913-555-5555","A dog! A panic in a pagoda!","A Toyota's a Toyota",
  Never odd or even","Rough Tough","Volatility","Ah, Satan sees Natasha","Guest Test","Rusty","McDonald's","Draw, o coward!","IHATEM3M3S---!!!",
  Lonely Tylenol","Metaverse with Google Fiber","Float","Take cover!","98752394875023","QWERTY","Leveled","HDDSSD","H2@@$@!@#(o00020320)"];
  let palindrome_array_answer: [bool; 30] = [true,false,false,true,false,true,false,true,false,false,true,true,
  true,false,false,true,false,false,true,false,true,false,false,false,false,false,false,false,false];
  let mut count_true: i32 = 0;
  let mut count_false: i32 = 0;
  println!("_");
  println!("Current phrase{:<18}Expected{:<4}Returned","","");
  println!("_");
  for i: usize in 0..30 {
    let is_palindrome: bool = find_palindrome(input: palindrome_array[i].to_string());
    println!("{:<32}{:<12}{}",palindrome_array[i],palindrome_array_answer[i],is_palindrome);
    if is_palindrome {count_true += 1;} else {count_false += 1;}
    assert_eq!(is_palindrome,palindrome_array_answer[i],"{} is not a palindrome!",palindrome_array[i]);
  }
  println!("_");
  println!("All words processed! There were {} palindromes and {} non-palindromes.",count_true,count_false);
}

```

```

in(){
  palindrome_verify_assert();
}

```