**ALGORITHM 2**

*public class PalindromeCheck {*

*public static void main(String[] args) {*

*String str = "madam";*

*boolean isPalindrome = isPalindrome(str);*

*System.out.println("Is the string a palindrome? " +*

*isPalindrome);*

*}*

*public static boolean isPalindrome(String str) {*

*int left = 0, right = str.length() - 1;*

*while (left < right) {*

*if (str.charAt(left) != str.charAt(right)) {*

*return false;*

*}*

*left++;*

*right--;*

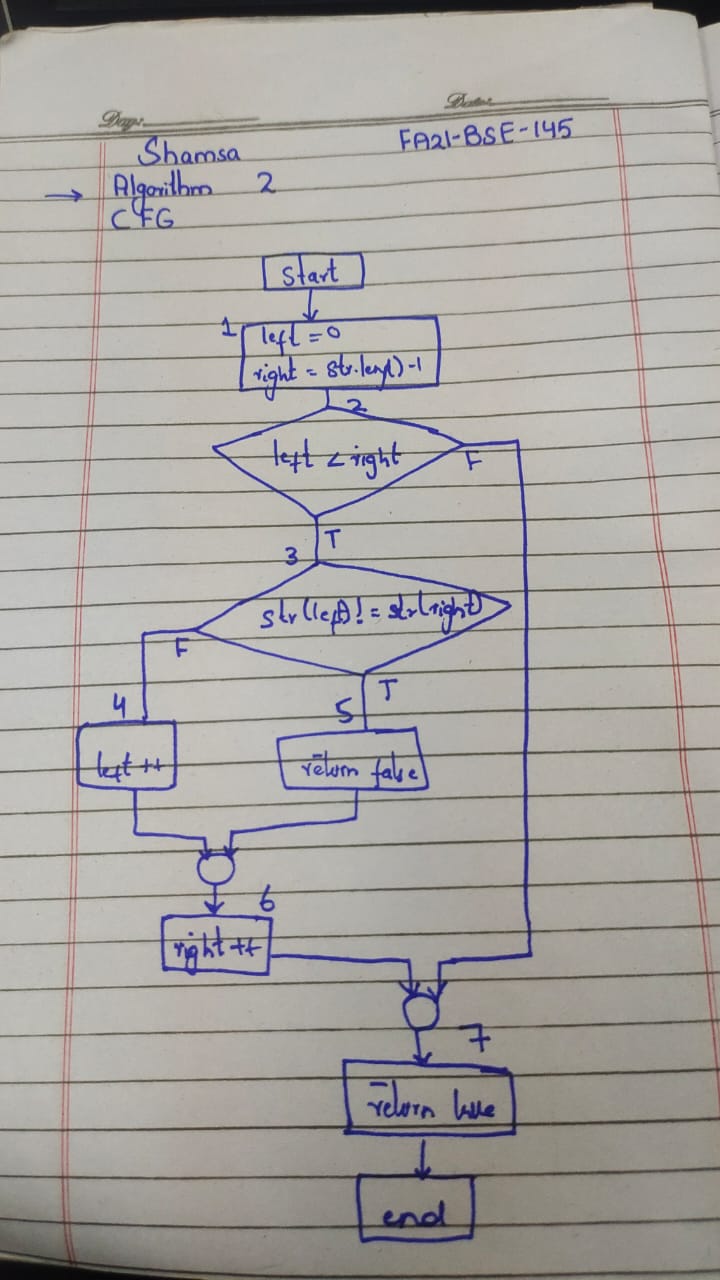
*}*

*return true;*

*}*

*}*

**CFG AND PATHS**

****

**A paper with writing on it

Description automatically generated**

**TEST CASES**

**ALGORITHM**

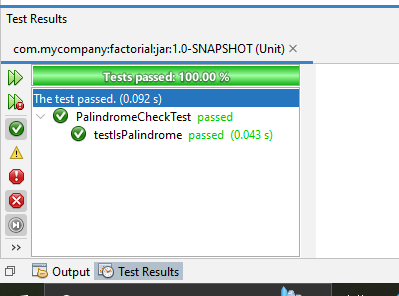
Check if a String is a Palindrome: Compares characters from the start

and end moving towards the center to check for equality.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TC-NO** | **DESCRIPTION** | **INPUT DATA** | **EXPECTED**  **OUTCOME** | **ACTUAL OUTCOME** | **STATUS** |
| TC-01 | Input correct String( left char is equal to right) | madam | palindrome | palindrome | pass |
| TC-02 | Input correct String | madam | palindrome | Not palindrome | Fail |
| TC-03 | Input incorrect string ( left char is not equal to right) | madan | Not palindrome | Not palindrome | pass |
| TC-04 | Input incorrect string | madan | Not palindrome | Palindrome | Fail |
| TC-05 | Input only one character | m | palindrome | palindrome | pass |
| TC-06 | Input empty string |  | palindrome | palindrome | Pass |
| TC-07 | Input one upper case letter | Madam | Not palindrome | Not  palindrome | Pass |

NOTE

* Test case 7 is not handled in code
* Every test case is covering paths
* Only pass test cases are executed in code



A screenshot of a computer

Description automatically generated