

## Project Documentation

### Machine Problem

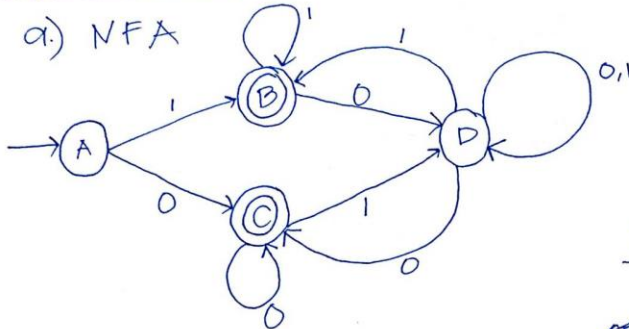
According to Merriam-Webster Dictionary, a *palindrome* comes from the Greek word *palindromes*, meaning “running back again”. It is a word, verse, number, or sentence that reads the same backward or forwards. Racecar and madam are a few famous examples of what palindromes are, they read the same even when they are reversed.

The machine problem requires a program that identifies whether a string is a palindrome or not. In addition, the program accepts only 0 and 1 as alphabets, any none binary found within the string is rejected or labeled as “not a palindrome”. The program must allow the user to input a string, then check if it is a palindrome or not by using loops to iterate between two indexes that compare the two for equality as well as check if it is 0 and 1. After checking, the program must output if the string is a palindrome or not.

### Requirements

#### Requirements

a) NFA



b) Components

$$Q = \{A, B, C, D\}$$

$$\Sigma = \{0, 1\}$$

$$S = \{A\}$$

$$F = \{B, C\}$$

#### TRANSITION

	0	1
A	C	B
B	D	B
C	C	D
D	C, D	B, D

## Source Code

```
let stat = "";
let string = "";
let palindrome = true;
let clear = document.getElementById("clear");
let strings = document.getElementById("str");
let result = document.getElementById("stat");

const validate = () => {
  string = document.getElementById("string").value;
  n = string.length;

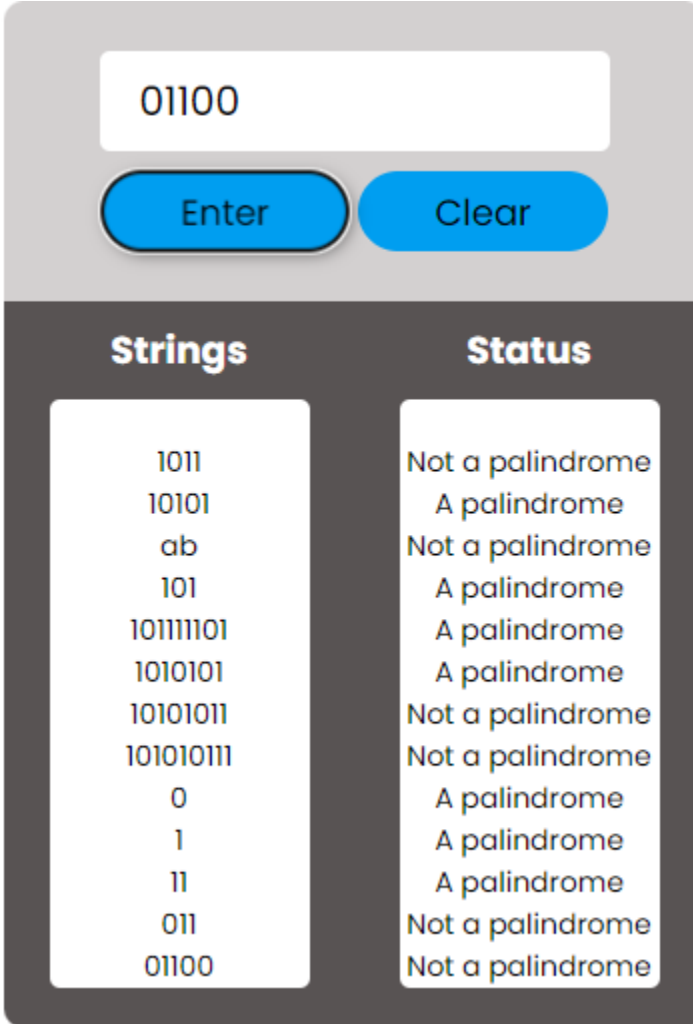
  if (string == "") {
    alert("No string found.");
  } else {
    for (let i = 0; i < Math.floor(n / 2); i++) {
      if (string[i] == "1" || string[i] == "0") {
        if (string[i] != string[n - i - 1]) {
          palindrome = false;
          break;
        }
      } else {
        palindrome = false;
        break;
      }
    }
    checkStatus();
  }
};

const checkStatus = () => {
  if (palindrome == true) {
    stat = "A palindrome";
  } else {
    stat = "Not a palindrome";
  }
  palindrome = true;
  strings.innerHTML += "\n" + string;
  result.innerHTML += "\n" + stat;
};

const clean = () => {
  strings.innerHTML = "";
  result.innerHTML = "";
};
```

```
};
```

### Program Output



Strings	Status
1011	Not a palindrome
10101	A palindrome
ab	Not a palindrome
101	A palindrome
101111101	A palindrome
1010101	A palindrome
10101011	Not a palindrome
101010111	Not a palindrome
0	A palindrome
1	A palindrome
11	A palindrome
011	Not a palindrome
01100	Not a palindrome

### Discussion

The following concepts and theories were used:

- Variables and Data Types
  - Boolean is used to indicate whether the string is a palindrome or not.
  - Strings were used to store input text.
- Control Structures
  - If and else statements were used to check the validity of the string.

- Loops
  - For loop is used to iterate through the string to check whether the two indexes are equal.
- Document Object Model
  - Used to interact with the HTML document.
- Functions
  - Functions were used to bundle codes to different components to perform a specific task within the program.

### **Solution**

Checking a string if it is a palindrome or not is rather simple since I went through it a couple of times yet in this problem, there are constraints as this is a machine that recognizes only 0s and 1s as its input which adds a little bit of difficulty.

I tackled the problem by first analyzing how I can efficiently iterate through the input entered by the user. I used a for loop that iterates halfway through the input and utilized two indexes to check both ends if they are equal. Each iteration compares the next index to the previous last index making it smaller up to the middle. The program will run in  $O(n/2)$  which I think is efficient enough rather than iterating throughout the string.

If the program finds any none binary element within the string, a Boolean variable will be assigned as false which indicates that it is not a palindrome. Lastly, it will display the string entered as well as the result of whether the string is a palindrome or not.