freeCodeCamp(🔥)

# JavaScript Algorithms and Data Structures – 5 Live Projects

**Name of the Student:** Gibran Khan Tareen
**Contact Details:** gibrantareen@gmail.com
**Github Profile**: @gibrankhantareen
**Linkedin:** @gibrankhantareen
**Institution:** Lovely Professional University
**Program:** BTech Computer Science & Engg.

(**Github Project Link**: Click Here)

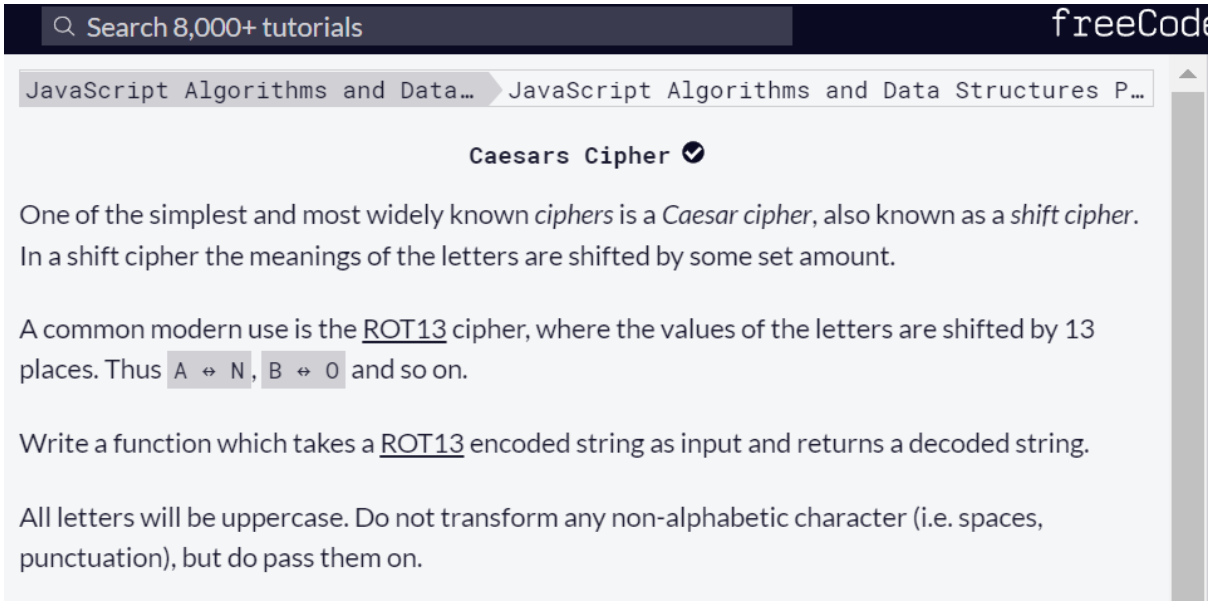
# Project 1 : Caesars Cipher

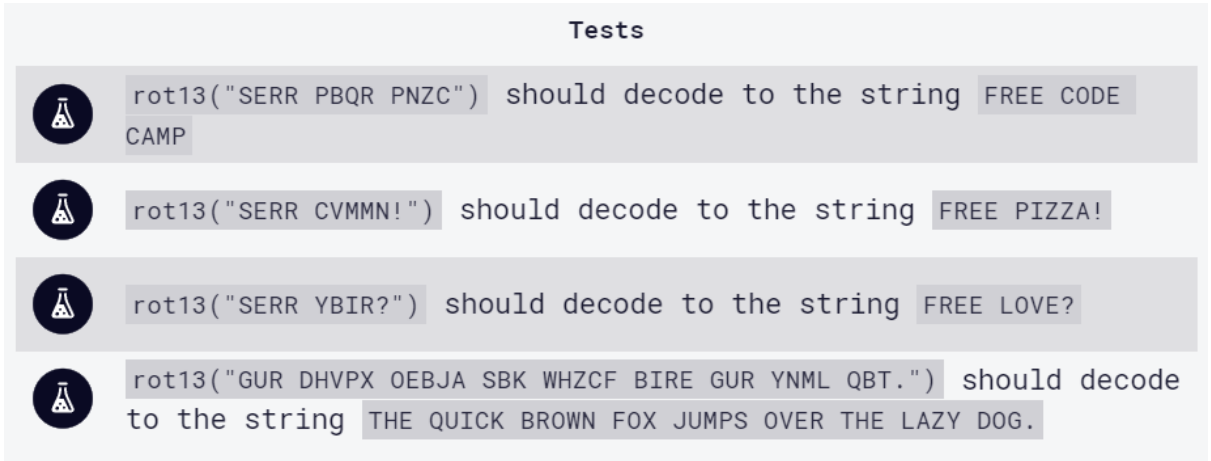
One of the simplest and most widely known *ciphers* is a *Caesar cipher*, also known as a *shift cipher*. In a shift cipher the meanings of the letters are shifted by some set amount.

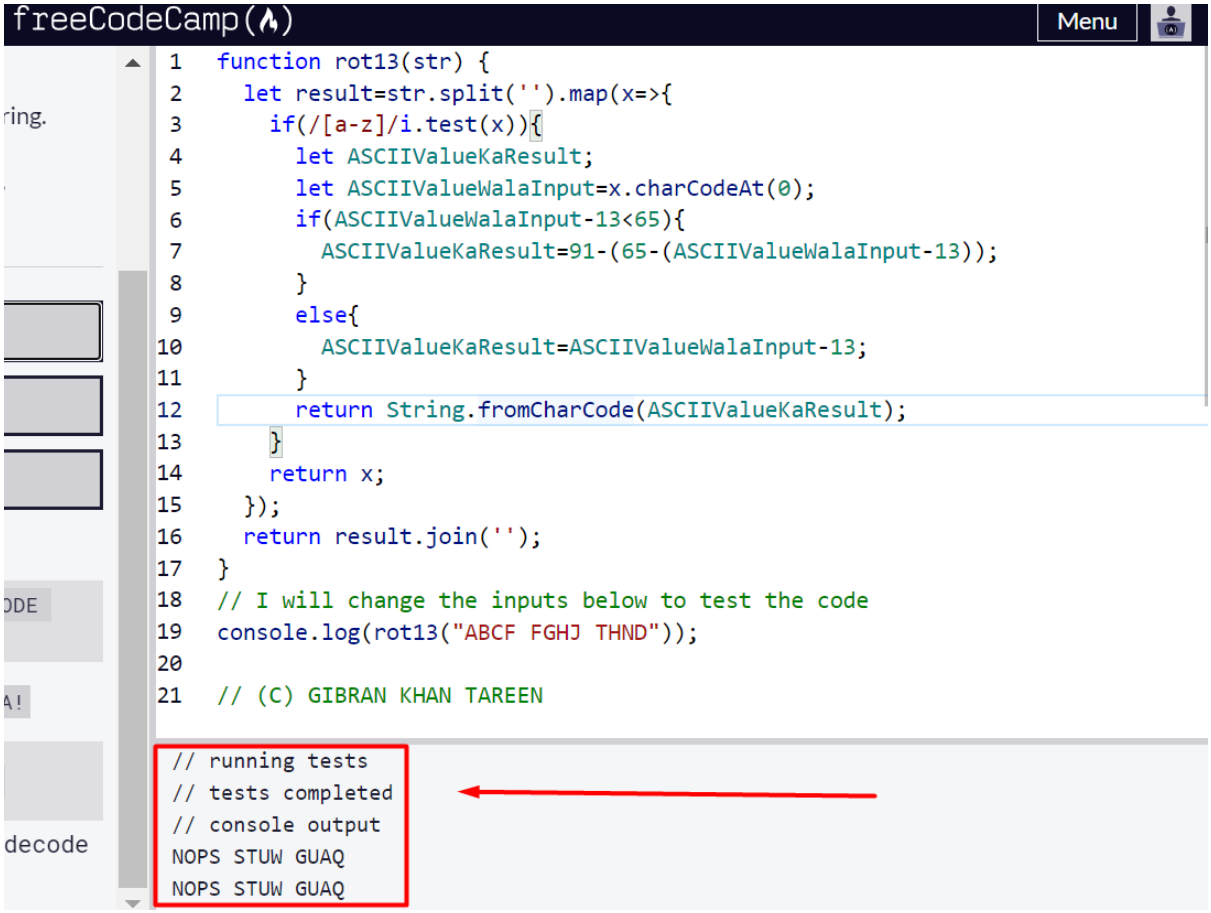
A common modern use is the ROT13 cipher, where the values of the letters are shifted by 13 places. Thus `A ↔ N`, `B ↔ O` and so on.

Write a function which takes a ROT13 encoded string as input and returns a decoded string.

All letters will be uppercase. Do not transform any non-alphabetic character (i.e. spaces, punctuation), but do pass them on.






## Hence Project #1 Completed ✔

# Project 2 : Telephone Number Validator

Return `true` if the passed string looks like a valid US phone number.

The user may fill out the form field any way they choose as long as it has the format of a valid US number. The following are examples of valid formats for US numbers (refer to the tests below for other variants):

```
555-555-5555
(555)555-5555
(555) 555-5555
555 555 5555
5555555555
1 555 555 5555
```

For this challenge you will be presented with a string such as `800-692-7753` or `8oo-six427676;laskdjf`. Your job is to validate or reject the US phone number based on any combination of the formats provided above. The area code is required. If the country code is provided, you must confirm that the country code is `1`. Return `true` if the string is a valid US phone number; otherwise return `false`.

**Tests**

- `telephoneCheck("555-555-5555")` should return a boolean.
- `telephoneCheck("1 555-555-5555")` should return `true`.
- `telephoneCheck("1 (555) 555-5555")` should return `true`.
- `telephoneCheck("5555555555")` should return `true`.
- `telephoneCheck("555-555-5555")` should return `true`.
- `telephoneCheck("(555)555-5555")` should return `true`.
- `telephoneCheck("1(555)555-5555")` should return `true`.
- `telephoneCheck("555-5555")` should return `false`.

- `telephoneCheck("1 555)555-5555")` should return `false`.
- `telephoneCheck("1 555 555 5555")` should return `true`.
- `telephoneCheck("1 456 789 4444")` should return `true`.
- `telephoneCheck("123**&!!asdf#")` should return `false`.
- `telephoneCheck("55555555")` should return `false`.
- `telephoneCheck("(6054756961)")` should return `false`.
- `telephoneCheck("2 (757) 622-7382")` should return `false`.
- `telephoneCheck("0 (757) 622-7382")` should return `false`.
- `telephoneCheck("-1 (757) 622-7382")` should return `false`.
- `telephoneCheck("2 757 622-7382")` should return `false`.
- `telephoneCheck("10 (757) 622-7382")` should return `false`.

- `telephoneCheck("27576227382")` should return `false`.
- `telephoneCheck("(275)76227382")` should return `false`.
- `telephoneCheck("2(757)6227382")` should return `false`.
- `telephoneCheck("2(757)622-7382")` should return `false`.
- `telephoneCheck("555)-555-5555")` should return `false`.
- `telephoneCheck("(555-555-5555")` should return `false`.
- `telephoneCheck("(555)5(55?)-5555")` should return `false`.
- `telephoneCheck("55 55-55-555-5")` should return `false`.
- `telephoneCheck("11 555-555-5555")` should return `false`.


## Hence Project #2 Completed ✔

# Cash Register

Design a cash register drawer function `checkCashRegister()` that accepts purchase price as the first argument (`price`), payment as the second argument (`cash`), and cash-in-drawer (`cid`) as the third argument.

`cid` is a 2D array listing available currency.

The `checkCashRegister()` function should always return an object with a `status` key and a `change` key.

Return `{status: "INSUFFICIENT_FUNDS", change: []}` if cash-in-drawer is less than the change due, or if you cannot return the exact change.

Return `{status: "CLOSED", change: [...]}` with cash-in-drawer as the value for the key `change` if it is equal to the change due.

Otherwise, return `{status: "OPEN", change: [...]}`, with the change due in coins and bills, sorted in highest to lowest order, as the value of the `change` key.

| Currency Unit | Amount |
|---|---|
| Penny | $0.01 (PENNY) |
| Nickel | $0.05 (NICKEL) |
| Dime | $0.1 (DIME) |
| Quarter | $0.25 (QUARTER) |
| Dollar | $1 (ONE) |
| Five Dollars | $5 (FIVE) |

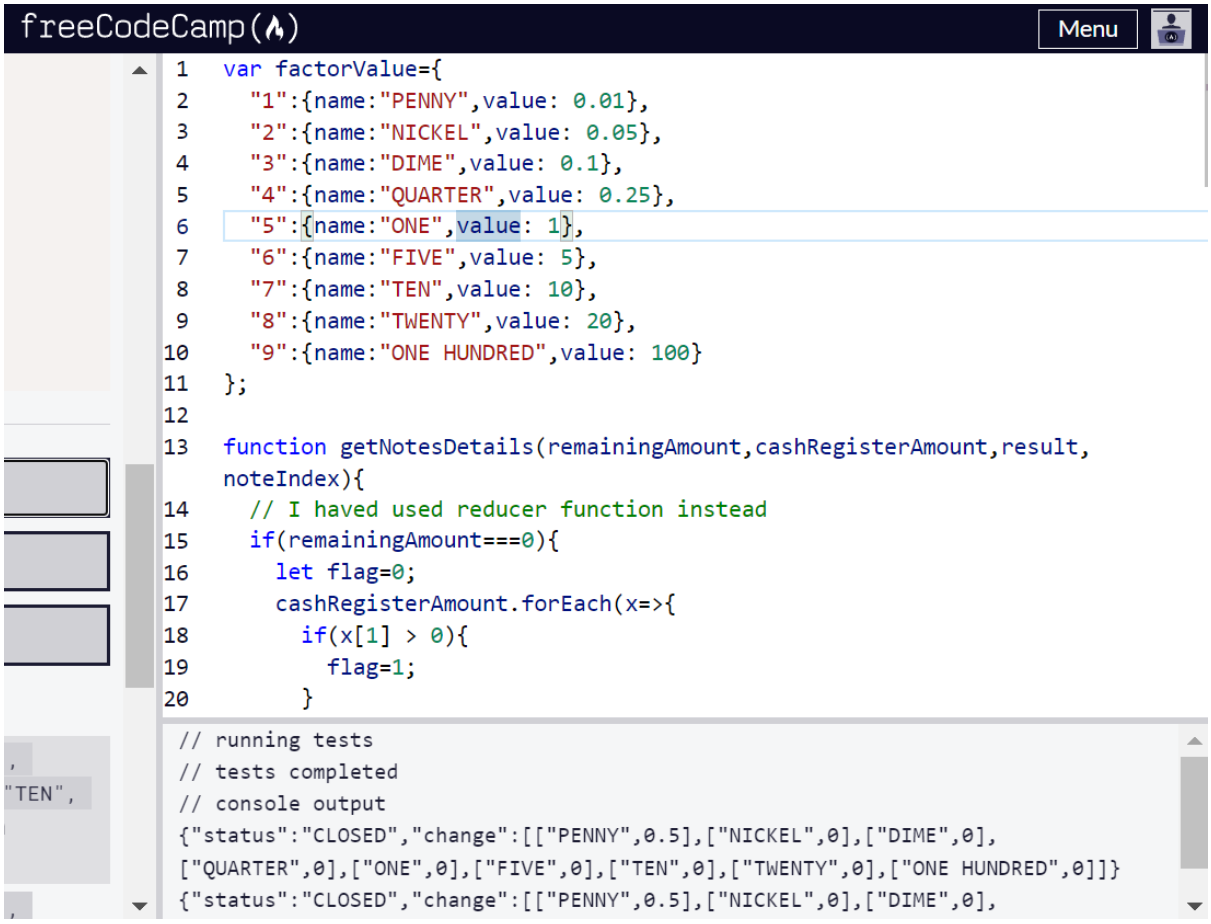
| | |
|---|---|
| Ten Dollars | $10 (TEN) |
| Twenty Dollars | $20 (TWENTY) |
| One-hundred Dollars | $100 (ONE HUNDRED |

See below for an example of a cash-in-drawer array:

```
[
  ["PENNY", 1.01],
  ["NICKEL", 2.05],
  ["DIME", 3.1],
  ["QUARTER", 4.25],
  ["ONE", 90],
  ["FIVE", 55],
  ["TEN", 20],
  ["TWENTY", 60],
  ["ONE HUNDRED", 100]
]
```

**Tests**

- `checkCashRegister(19.5, 20, [["PENNY", 1.01], ["NICKEL", 2.05], ["DIME", 3.1], ["QUARTER", 4.25], ["ONE", 90], ["FIVE", 55], ["TEN", 20], ["TWENTY", 60], ["ONE HUNDRED", 100]])` should return an object.
- `checkCashRegister(19.5, 20, [["PENNY", 1.01], ["NICKEL", 2.05], ["DIME", 3.1], ["QUARTER", 4.25], ["ONE", 90], ["FIVE", 55], ["TEN", 20], ["TWENTY", 60], ["ONE HUNDRED", 100]])` should return `{status: "OPEN", change: [["QUARTER", 0.5]]}`.
- `checkCashRegister(3.26, 100, [["PENNY", 1.01], ["NICKEL", 2.05], ["DIME", 3.1], ["QUARTER", 4.25], ["ONE", 90], ["FIVE", 55], ["TEN", 20], ["TWENTY", 60], ["ONE HUNDRED", 100]])` should return `{status: "OPEN", change: [["TWENTY", 60], ["TEN", 20], ["FIVE", 15], ["ONE", 1], ["QUARTER", 0.5], ["DIME", 0.2], ["PENNY", 0.04]]}`.
- `checkCashRegister(19.5, 20, [["PENNY", 0.01], ["NICKEL", 0], ["DIME", 0], ["QUARTER", 0], ["ONE", 0], ["FIVE", 0], ["TEN", 0], ["TWENTY", 0], ["ONE HUNDRED", 0]])` should return `{status: "INSUFFICIENT_FUNDS", change: []}`.
- `checkCashRegister(19.5, 20, [["PENNY", 0.01], ["NICKEL", 0], ["DIME", 0], ["QUARTER", 0], ["ONE", 1], ["FIVE", 0], ["TEN", 0], ["TWENTY", 0], ["ONE HUNDRED", 0]])` should return `{status: "INSUFFICIENT_FUNDS", change: []}`.




***(complete source code given in the folder)***

## Hence Project #3 Completed ✔

# Project 4 : Palindrome Checker

Return `true` if the given string is a palindrome. Otherwise, return `false`.

A *palindrome* is a word or sentence that's spelled the same way both forward and backward, ignoring punctuation, case, and spacing.

**Note:** You'll need to remove **all non-alphanumeric characters** (punctuation, spaces and symbols) and turn everything into the same case (lower or upper case) in order to check for palindromes.

We'll pass strings with varying formats, such as `racecar`, `RaceCar`, and `race CAR` among others.

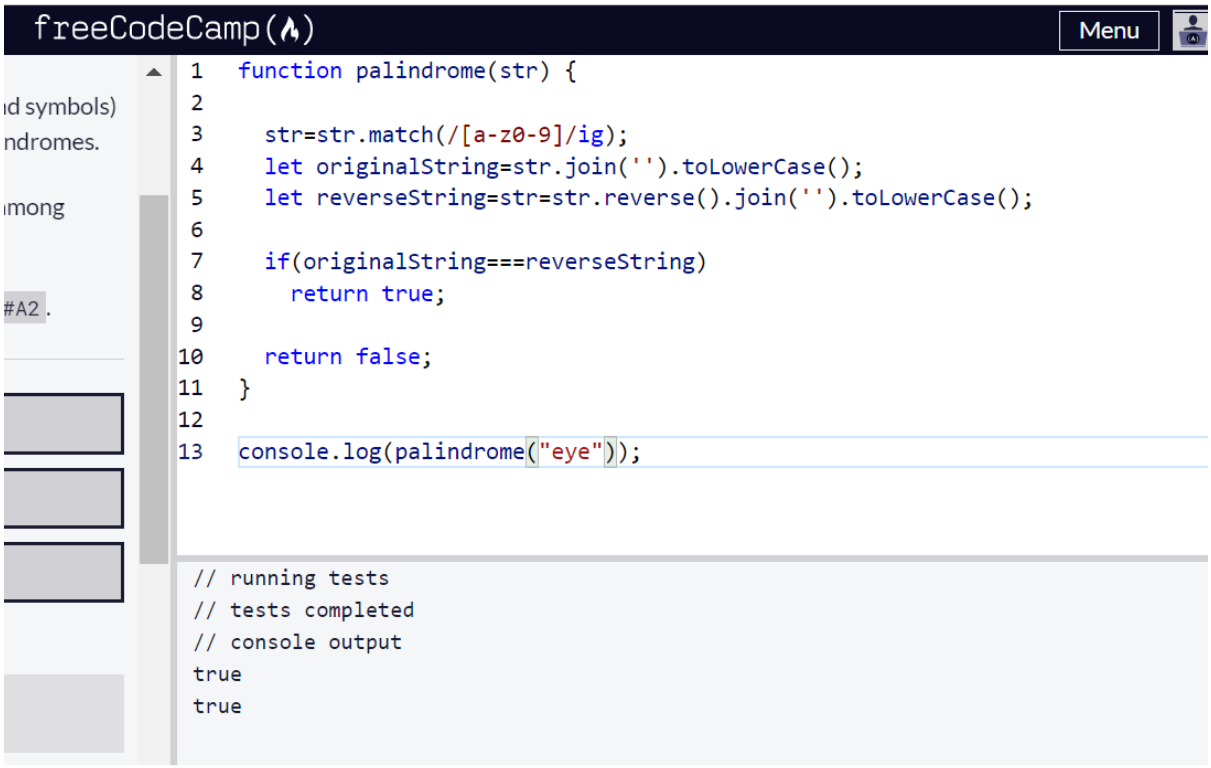
We'll also pass strings with special symbols, such as `2A3*3a2`, `2A3 3a2`, and `2_A3*3#A2`.

**Tests**

- `palindrome("eye")` should return a boolean.
- `palindrome("eye")` should return `true`.
- `palindrome("_eye")` should return `true`.
- `palindrome("race car")` should return `true`.
- `palindrome("not a palindrome")` should return `false`.
- `palindrome("A man, a plan, a canal. Panama")` should return `true`.
- `palindrome("never odd or even")` should return `true`.
- `palindrome("nope")` should return `false`.

**(Please Turn Over)**

- `palindrome("nope")` should return `false`.
- `palindrome("almostomla")` should return `false`.
- `palindrome("My age is 0, 0 si ega ym.")` should return `true`.
- `palindrome("1 eye for of 1 eye.")` should return `false`.
- `palindrome("0_0 (: /-\ :) 0-0")` should return `true`.
- `palindrome("five|\_/|four")` should return `false`.




## Hence Project #4 Completed ✔

# Roman Numeral Converter

Convert the given number into a roman numeral.

| Roman numerals | Arabic numerals |
|---|---|
| M | 1000 |
| CM | 900 |
| D | 500 |
| CD | 400 |
| C | 100 |
| XC | 90 |
| L | 50 |
| XL | 40 |
| X | 10 |
| IX | 9 |
| V | 5 |
| IV | 4 |
| I | 1 |

All roman numerals answers should be provided in upper-case.

**Tests**

- `convertToRoman(2)` should return the string `II`.
- `convertToRoman(3)` should return the string `III`.
- `convertToRoman(4)` should return the string `IV`.
- `convertToRoman(5)` should return the string `V`.
- `convertToRoman(9)` should return the string `IX`.
- `convertToRoman(12)` should return the string `XII`.
- `convertToRoman(16)` should return the string `XVI`.
- `convertToRoman(29)` should return the string `XXIX`.
- `convertToRoman(44)` should return the string `XLIV`.
- `convertToRoman(45)` should return the string `XLV`.

freeCodeCamp Menu

```
var factorValue={
  "1":{numeric:1,roman: 'I'},
  "2":{numeric:4,roman: 'IV'},
  "3":{numeric:5,roman: 'V'},
  "4":{numeric:9,roman: 'IX'},
  "5":{numeric:10,roman: 'X'},
  "6":{numeric:40,roman: 'XL'},
  "7":{numeric:50,roman: 'L'},
  "8":{numeric:90,roman: 'XC'},
  "9":{numeric:100,roman: 'C'},
  "10":{numeric:400,roman: 'CD'},
  "11":{numeric:500,roman: 'D'},
  "12":{numeric:900,roman: 'CM'},
  "13":{numeric:1000,roman: 'M'},
};
function convertToRoman(num) {
    let result = '';
    for (let i= 13; i>0; i--) {
        while (factorValue[i]["numeric"] <= num) {
            result += factorValue[i]["roman"];
            num -= factorValue[i]["numeric"];
        }
    }
    return result;
}                                   // (C) GIBRAN KHAN TAREEN
console.log(convertToRoman(83));
```

freeCodeCamp Menu

```
};
function convertToRoman(num) {
    let result = '';
    for (let i= 13; i>0; i--) {
        while (factorValue[i]["numeric"] <= num) {
            result += factorValue[i]["roman"];
            num -= factorValue[i]["numeric"];
        }
    }
    return result;
}                                   // (C) GIBRAN KHAN TAREEN
console.log(convertToRoman(83));
```

```
// running tests
LXXXIII
// tests completed
// console output
LXXXIII
```

**Hence All Projects 5/5 Completed** ✔

After successful submission, I got the certificate