**Global Trend Programming Profile Assessment**

**Assignment Submission**

**Question 1:** Write a function that prints the numbers from 1 to 100. But for multiples of three, print "Fizz" instead of the number, and for the multiples of five, print "Buzz". For numbers that are multiples of both three and five, print "FizzBuzz".

**Code:**

*const* fizzBuzz = () *=>* {

*let* output = "";

      for (*let* i = 1; i <= 100; i++) {

        if (i % 3 === 0) output += "Fizz";

        if (i % 5 === 0) output += "Buzz";

        if (i % 3 && i % 5) output += i;

        output += "\n";

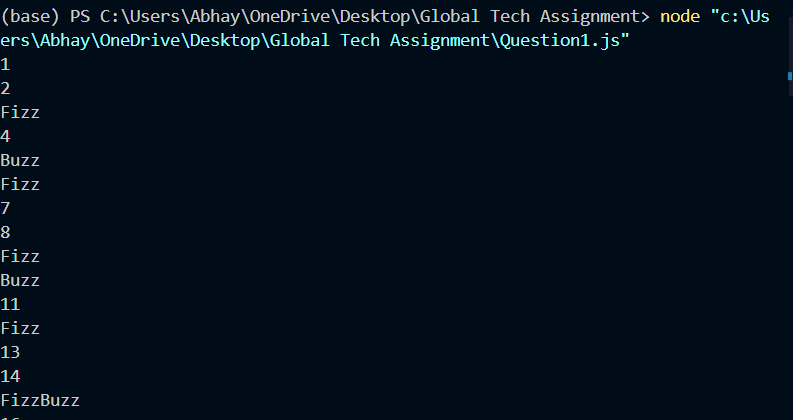
        }

        console.log(output);

    };

    fizzBuzz();

**Output :**



**Question 2:** Write a function that takes a string input representing a simple arithmetic expression (only addition and subtraction) and returns the result.

**Code:**

/\*

Write a function that takes a string input representing a simple arithmetic expression (only addition and subtraction) and returns the result.

Example:

Input: "1+2+3-4"

Output: 2

\*/

*const* arithmetic = (*str*) *=>* {

*let* output = 0;

*let* lastSign = 1,

        curr = 0;

      for (*let* i = 0; i < *str*.length; i++) {

        if (*str*[i] === "+" || *str*[i] == "-") {

          output += lastSign \* curr;

          curr = 0;

          lastSign = *str*[i] === "+" ? 1 : -1;

        } else {

          curr = curr \* 10 + parseInt(*str*[i]);

        }

      }

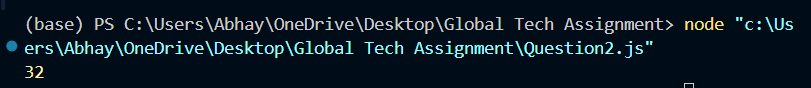
      output += lastSign \* curr;

      return output;

    };

    console.log(arithmetic("1+42+3-14")); // 32

**Output :**

****

**Question 3:** Write a function that takes a nested array and returns a flattened array.

**Code:**

/\*Write a function that takes a nested array and returns a flattened array.\*/

*const* flatten = (*arr*) *=>* {

  return *arr*.reduce(

    (*acc*, *val*) *=>* *acc*.concat(*Array*.isArray(*val*) ? flatten(*val*) : *val*),

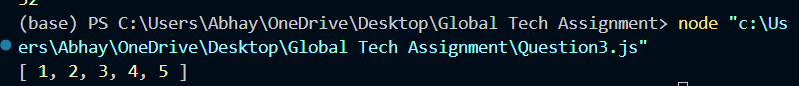
    []

  );

};

console.log(flatten([1, [2, [3, 4], 5]])); // [1, 2, 3, 4, 5]

**Output :**

****

**Question 4:** Write a function that checks if two given strings are anagrams of each other.

**Code:**

/\*Write a function that checks if two given strings are anagrams of each other.

 \*/

*const* isAnagram = (*str1*, *str2*) *=>* {

  // initialise an array of 52 elements to store the count of each letter

*const* letterHash = new *Array*(52);

  letterHash.fill(0);

  // Upadate letterhash with the first string

  for (*const* c of *str1*) {

*const* index =

      c >= "a"

        ? c.charCodeAt(0) - "a".charCodeAt(0)

        : c.charCodeAt(0) - "A".charCodeAt(0) + 26;

    letterHash[index]++;

  }

  // Update letterhash with the second string

  for (*const* c of *str2*) {

*const* index =

      c >= "a"

        ? c.charCodeAt(0) - "a".charCodeAt(0)

        : c.charCodeAt(0) - "A".charCodeAt(0) + 26;

    letterHash[index]--;

  }

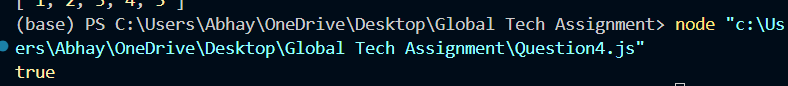
  // Check if the letterHash is all zeros

  return letterHash.every((*val*) *=>* *val* === 0);

};

console.log(isAnagram("Global", "lobGal")); // true

**Output:**

****

**Question 5:** Write a function that takes an array and returns a new array with duplicates removed.

**Code:**

/\*Write a function that takes an array and returns a new array with duplicates removed.\*/

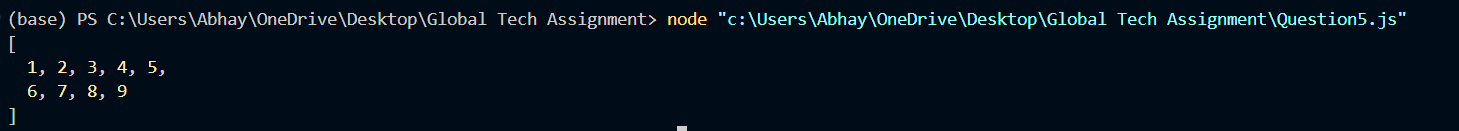
*const* removeDuplicates = (*arr*) *=>* {

  return *Array*.from(new *Set*(*arr*));

};

console.log(removeDuplicates([1, 2, 3, 4, 5, 5, 6, 7, 8, 9, 9])); // [1, 2, 3, 4, 5, 6, 7, 8, 9]

**Output :**

****

**Question 6:** Write a function that takes a string and capitalizes the first letter of each word in the string

**Code:**

/\*

Write a function that takes a string and capitalizes the first letter of each word in the string\*/

*const* capitaliseEachWord = (*str*) *=>* {

  return *str*

    .split(" ")

    .map((*word*) *=>* {

      return *word*.charAt(0).toUpperCase() + *word*.slice(1);

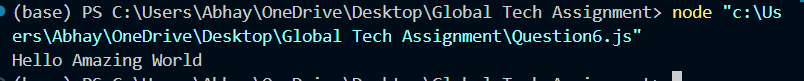
    })

    .join(" ");

};

console.log(capitaliseEachWord("hello amazing world")); // Hello Amazing World

**Output:**

****

**Question 7:** Write a function that generates the first n numbers of the Fibonacci sequence.

**Code:**

*const* fibonacci = (*n*) *=>* {

*const* series = [0, 1];

  for (*let* i = 2; i < *n*; i++) {

    series.push(series[i - 1] + series[i - 2]);

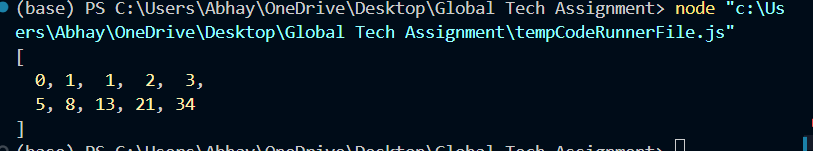
  }

  return series.slice(0, *n*);

};

console.log(fibonacci(10)); // [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]

**Output:**

****

**Question 8:**

Implement a simple HashMap class with put, get, and remove methods.

**Code:**

/\*Implement a simple HashMap class with put, get, and remove methods.\*/

*class* HashMap {

*constructor*() {

    this.map = {};

  }

  put(*key*, *value*) {

    this.map[*key*] = *value*;

  }

  get(*key*) {

    return this.map[*key*];

  }

  remove(*key*) {

    delete this.map[*key*];

  }

}

*const* map = new HashMap();

map.put("name", "John");

map.put("age", 30);

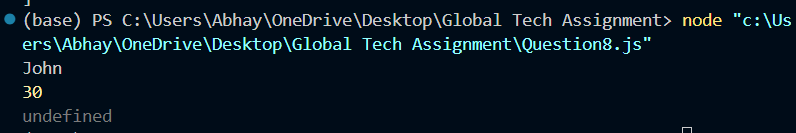
console.log(map.get("name")); // John

console.log(map.get("age")); // 30

map.remove("name");

console.log(map.get("name")); // undefined  (key "name" has been removed)

**Output:**

****

**Question 9:** Write a function that filters out even numbers from an array.

**Code:**

/\*Write a function that filters out even numbers from an array.\*/

*const* filterEvenNumbers = (*arr*) *=>* {

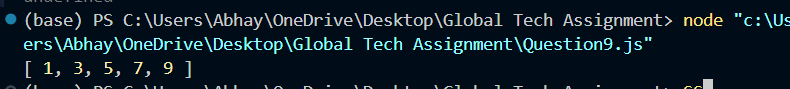
  return *arr*.filter((*num*) *=>* *num* % 2 !== 0);

};

*const* filtered = filterEvenNumbers([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]);

console.log(filtered); // [1, 3, 5, 7, 9]

**Output:**

****

**Question 10:**

Write a function that converts a given string to title case (capitalizing the first letter of each word).

**Code:**

/\* Write a function that converts a given string to title case (capitalizing the first letter of each word). \*/

*const* titleCase = (*str*) *=>* {

    return *str*

        .split(" ")

        .map((*word*) *=>* {

            return *word*.charAt(0).toUpperCase() + *word*.toLowerCase().slice(1);

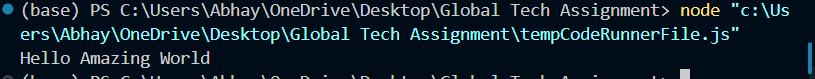
        })

        .join(" ");

};

console.log(titleCase("hello amaZiNg world")); // Hello Amazing World

**Output:**

****