Experiment 8

**Name** : Mohammad Wasi

**SAP ID** : 500110709

**Batch** : AIML B8

Aim: **: NodeJS basic exercises**..**.**

**Objective :**

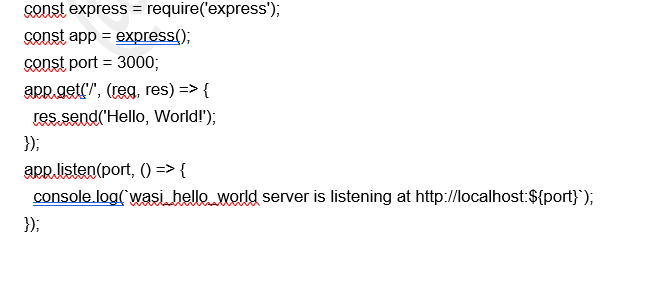
1. Hello World Server: Create an HTTP server using Node.js and Express to respond with "Hello, World!" to requests.
2. Regular Expression Replacement: Implement a Node.js program to replace two or more 'a's with 'b' in a string using a Regular Expression.
3. Basic Calculator: Develop an HTTP-based basic calculator using Node.js and Express for arithmetic operations.
4. Array Iteration: Write Node.js code to iterate over an array and print each element with its index.

Theory : +

Node.js enables server-side JavaScript programming. Express, a popular framework, simplifies building web servers. Regular Expressions provide powerful string manipulation tools. In HTTP-based applications, Express routes handle different endpoints. Functions like parseFloat convert string inputs to numbers for calculations. Array iteration using forEach is efficient for processing each element in an array..

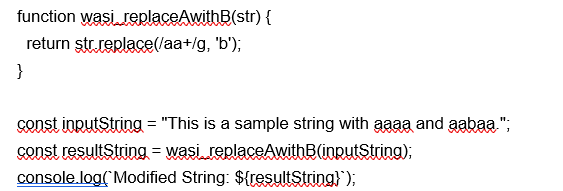
**Code**

**Question 1: Create a simple “Hello, World!” server using Node.js and Express.**



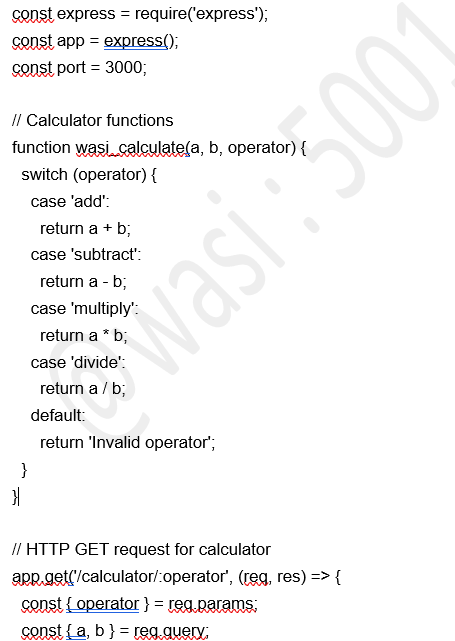


**Question 2: Write a node.js program to replace two or more a's with the letter b on the given string using Regular Expression.**

****



**Question 3: Create a basic calculator that can perform arithmetic operations (addition, subtraction, multiplication, and division) through HTTP requests.**



const result = wasi\_calculate(parseFloat(a), parseFloat(b), operator);

res.send(`Result: ${result}`);

});

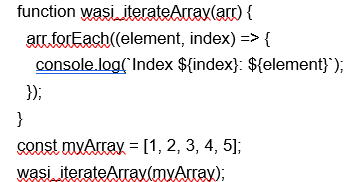
app.listen(port, () => {

console.log(`wasi\_calculator server is listening at http://localhost:${port}`);

});



**Question 4: Write a node.js code to iterate over the given array.**



Experiment 9

**Name** : Mohammad Wasi

**SAP ID** : 500110709

**Batch** : AIML B8

Aim: **Working with Data.**

**Objective :**

Connect to MongoDB: Establish a connection to a MongoDB database using NodeJS.

Student Details Storage: Develop an application to store and manage details of students in the MongoDB database.

Search Application: Implement a search functionality to find students based on specified criteria.

Shopping Center Application: Create an application for a shopping center with features to add, delete, update item details, generate stock reports, and manage sales.

Theory (50 words):

NodeJS, with its MongoDB driver, facilitates database operations. Storing student data involves defining a schema and using Mongoose to interact with MongoDB. Search applications use MongoDB queries to filter results. The shopping center application utilizes MongoDB for CRUD operations on items, providing efficient stock management and sales tracking.

Theory :

NodeJS, with its MongoDB driver, facilitates database operations. Storing student data involves defining a schema and using Mongoose to interact with MongoDB. Search applications use MongoDB queries to filter results. The shopping center application utilizes MongoDB for CRUD operations on items, providing efficient stock management and sales tracking.

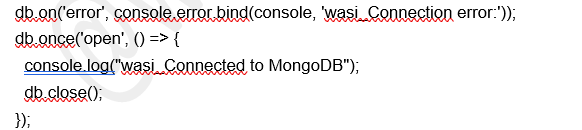
**Code**

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/mydb', { useNewUrlParser: true, useUnifiedTopology: true });

const db = mongoose.connection;

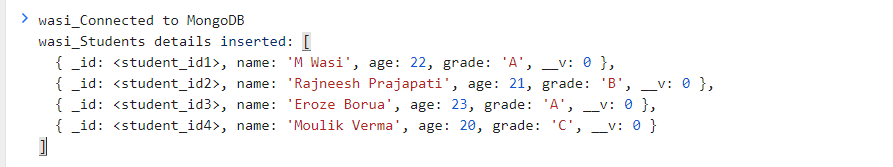
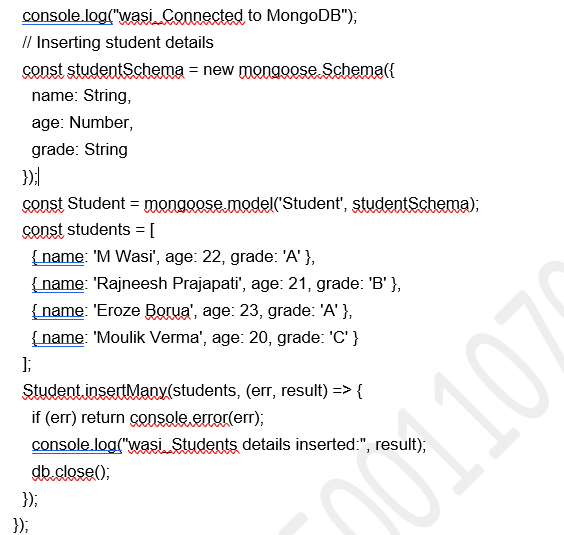
**Question 1: Create a NodeJS application to connect to a MongoDB database.**



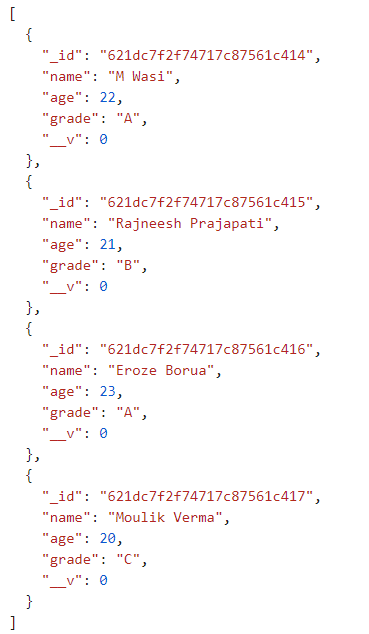
**Question 2: Create an application to store the details of students in a database.**

db.on('error', console.error.bind(console, 'wasi\_Connection error:'));

db.once('open', () => {

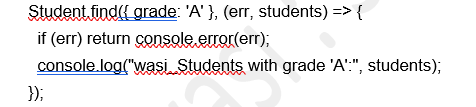


Console

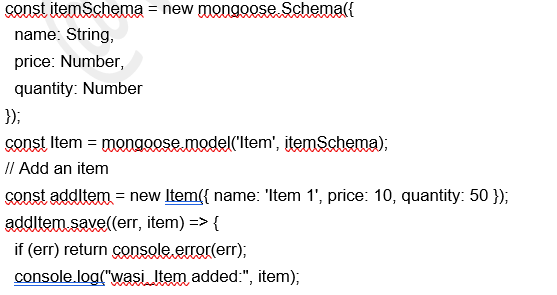


Database DB.json

**Question 3: Create a search application for finding the students based on given search criteria.**



**Question 4: Write a program to create an application for a shopping center.**

});

// Delete an item

Item.deleteOne({ name: 'Item 1' }, (err) => {

