```
const express = require('express');
const bodyParser = require('body-parser');

const app = express();
const port = 3007;


app.use(bodyParser.urlencoded({ extended: true }));


app.set('view engine', 'ejs');


app.get('/', (req, res) => {
  res.render('index');
});


app.post('/submit', (req, res) => {
  const name = req.body.name;
  const grade = req.body.grade;


  res.render('display', { name: name, grade: grade });
});


app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

Section A

Question 1

a. **Objective**: Extend the current code to perform the following tasks:

- Add input validation for the name and grade fields. Ensure that the name field is not empty, and the grade field must be a numeric value between 0 and 100.
- After validation, display an error message on the same page if the validation fails.
- Modify the POST /submit route to store the submitted data (name and grade) in an in-memory JavaScript array.

    - Add a new GET /results route that displays a list of all submitted names and grades from the array in an HTML table.

b. **Requirements**:

    - Add server-side validation in the POST /submit route to ensure the following:
        - The name field should not be empty.
        - The grade should be a number between 0 and 100.
        - If validation fails, the user should remain on the form page and see an appropriate error message.
    - Store the submitted name and grade in an array after passing validation.
    - Create a new route (/results) that retrieves and displays the list of all submitted names and grades in an HTML table using EJS.

Bonus:

- Modify the /results route to show grades in letter format (A, B, C, D, E, F) based on the provided grading scale:
    - 80-100: A
    - 70-79: B
    - 60-69: C
    - 50-59: D
    - 40-49: E
    - Below 40: F

Test Case:

- Input invalid name (e.g., empty string).
- Input invalid grade (e.g., negative number, string).
- Submit multiple valid records and view them in the /results table.

Question 2

a. Extend the above code to include a simple **registration** and **login** system using **Node.js** and **Express.js** with the following features:
    - Store user information (username, password) using an **in-memory array**
    - Ensure users cannot register with the same username.
    - Allow users to log in using the correct credentials and be redirected to a welcome page upon successful login.

b. **Requirements**:
    - **Registration Route (/register)**:
        - Create a GET route to display the registration form (fields: username, password, confirm password).
        - Create a POST route that validates the form input:
            - Check that the username is unique.
            - Check that the password and confirm password match.
            - Hash the password using bcrypt and store it in the database (or array).
        - If validation fails, show an appropriate error message on the same page.
    - **Login Route (/login)**:
        - Create a GET route to display the login form (fields: username, password).
        - Create a POST route to authenticate users:
            - Check if the username exists.
            - Use bcrypt to compare the hashed password with the entered password.
            - If authentication is successful, redirect the user to a welcome page.
            - If authentication fails, show an error message on the same page.
    - **Welcome Route (/welcome)**:
        - After a successful login, display a welcome page with a message like "Welcome, [username]!".

c. **Bonus**:
    - Implement a **logout** feature that clears the session data and redirects to the login page.

Test Case:

- Register with valid details (matching passwords).
- Try to register with an existing username and check for an error.
- Login with valid credentials.
- Try logging in with incorrect credentials and check for error handling.
- After a successful login, view the welcome page, and ensure the session remains active until the user logs out.

Section B

Fill-in-the-Blank Questions

1. In Express.js, to parse URL-encoded form data, we use the middleware _____________.
2. The GET method is used to retrieve data, while the _____________ method is used to send data to the server, often used for form submission.
3. To hash passwords in Node.js, the package _____________ is commonly used.
4. In Express.js, the command to start a server listening on a port is _____________.
5. The req.body object contains data sent in the body of a _____________ request.
6. In Express.js, the default view engine can be set using the command app.set('_____________', 'ejs');.
7. The method app.use() is used to register middleware functions in the _____________ lifecycle.
8. The app.listen(port, callback) function is used to start a server, where _____________ is the function executed once the server starts listening.
9. The function _____________ in Express.js is used to render dynamic content using a templating engine like EJS.
10. In Express.js, a route defined for handling HTTP requests at the / path with the GET method looks like app._____________('/', callback).

Section C: Multiple Choice Questions

1. **Which of the following middleware is used to parse incoming request bodies in Express.js?**
   - a) body-parser
   - b) cookie-parser
   - c) url-parser
   - d) static-parser
2. **What is the default port on which an Express.js application runs if not specified?**
   - a) 3000
   - b) 8080
   - c) 8000
   - d) 5000
3. **Which method is used to send a response back to the client in Express.js?**
   - a) req.send()
   - b) res.write()

  - c) res.send()
  - d) req.response()

4. **In Express.js, the route parameter :id in /users/:id will be available in which object?**
   - a) req.body
   - b) req.params
   - c) req.query
   - d) req.route
5. **Which of the following is a valid way to redirect a user to a different page in Express.js?**
   - a) res.sendRedirect('/home')
   - b) res.location('/home')
   - c) res.redirect('/home')
   - d) res.go('/home')
6. **Which of the following Node.js module is used to work with the file system?**
   - a) fs
   - b) os
   - c) http
   - d) path
7. **What is the purpose of using bcrypt in a Node.js application?**
   - a) To compress files
   - b) To secure sessions
   - c) To hash passwords
   - d) To encrypt routes
8. **Which of the following commands is used to install a package in Node.js?**
   - a) npm download <package_name>
   - b) npm install <package_name>
   - c) npm get <package_name>
   - d) npm use <package_name>
9. **In Express.js, which method is used to serve static files like images or CSS?**
   - a) app.serve()
   - b) app.static()
   - c) app.use(express.static())

   - d) app.load()

10. **Which command initializes a new Node.js project by creating a package.json file?**
    - a) npm start
    - b) npm init
    - c) node init
    - d) npm create