

Question1:

The screenshot shows the VS Code editor with a file named `q1.js` open. The code implements a `lowerCaseWords` function that takes a `mixedArray` and returns a Promise. The function checks if the input is an array; if not, it rejects with an error. If it is an array, it filters for strings and maps them to lowercase. The `mixedArray` is defined as `["PIZZA", 10, true, 25, false, "Wings"]`. The terminal shows the execution of `node q1.js`, which outputs `['pizza', 'wings']`.

```
JS q1.js U question-1/q1.js/...
1  function lowerCaseWords(mixedArray) {
2      return new Promise((resolve, reject) => {
3          if (!Array.isArray(mixedArray)) {
4              return reject(new Error("Input must be an array"));
5          }
6
7          const lowerCased = mixedArray
8              .filter(item => typeof item === 'string')
9              .map(str => str.toLowerCase());
10
11         resolve(lowerCased);
12     });
13 }
14
15 const mixedArray = ["PIZZA", 10, true, 25, false, "Wings"];
16 lowerCaseWords(mixedArray)
17     .then(result => console.log(result))
18     .catch(error => console.error(error));
```

TERMINAL

```
MacBook-Pro-5:question-1 mericyassine$ clear
MacBook-Pro-5:question-1 mericyassine$ node q1.js
[ 'pizza', 'wings' ]
MacBook-Pro-5:question-1 mericyassine$
```

Question 2:

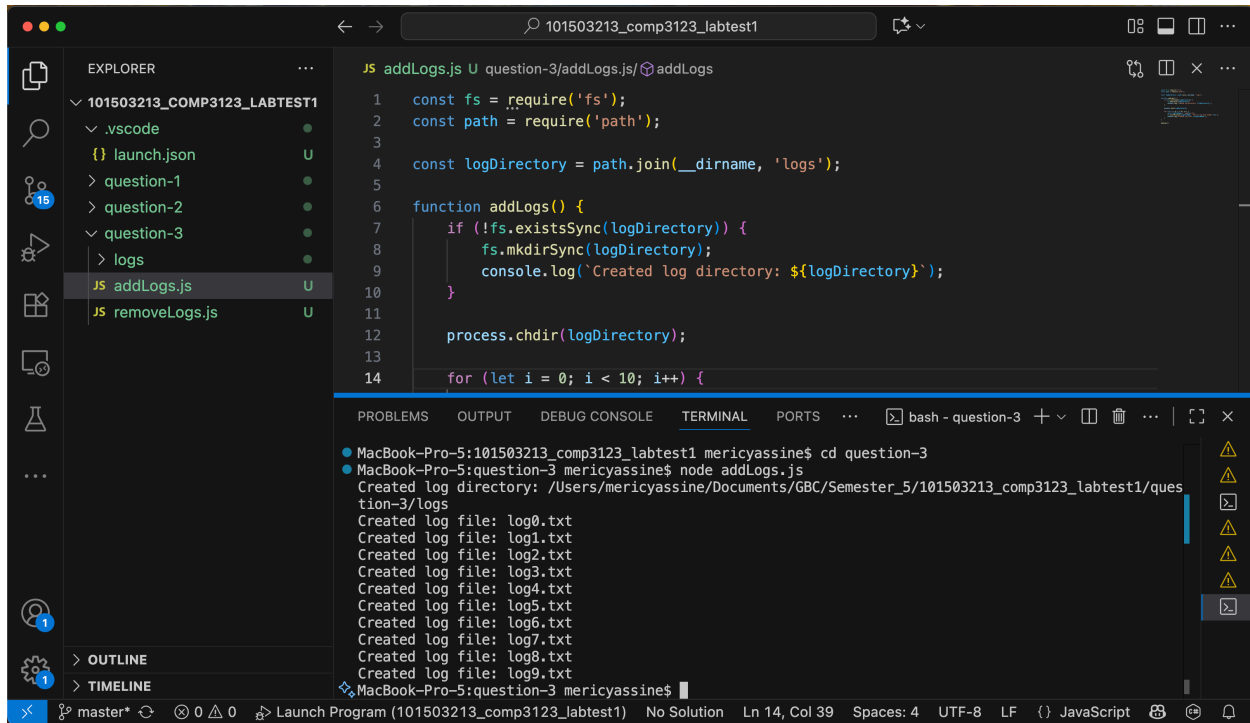
The screenshot shows the VS Code editor with a file named `q2.js` open. The code implements a `rejectedPromise` function that returns a Promise which rejects with a delayed exception. It also shows the `resolvedPromise` and `rejectedPromise` functions being called. The terminal shows the execution of `node q2.js`, which outputs `{ message: 'delayed success!' }` and `{ error: 'delayed exception!' }`.

```
JS q2.js U question-2/q2.js/...
7  }
8
9  const rejectedPromise = () => {
10     return new Promise( (_, reject) => {
11         setTimeout(() => {
12             reject({ 'error': 'delayed exception!' });
13         }, 500);
14     });
15 }
16
17 resolvedPromise()
18     .then(result => console.log(result))
19     .catch(error => console.error(error));
20
21 rejectedPromise()
22     .then(result => console.log(result))
23     .catch(error => console.error(error));
24
```

TERMINAL

```
MacBook-Pro-5:question-1 mericyassine$ clear
MacBook-Pro-5:101503213_comp3123_labtest1 mericyassine$ cd question-2
MacBook-Pro-5:question-2 mericyassine$ node q2.js
{ message: 'delayed success!' }
{ error: 'delayed exception!' }
MacBook-Pro-5:question-2 mericyassine$
```

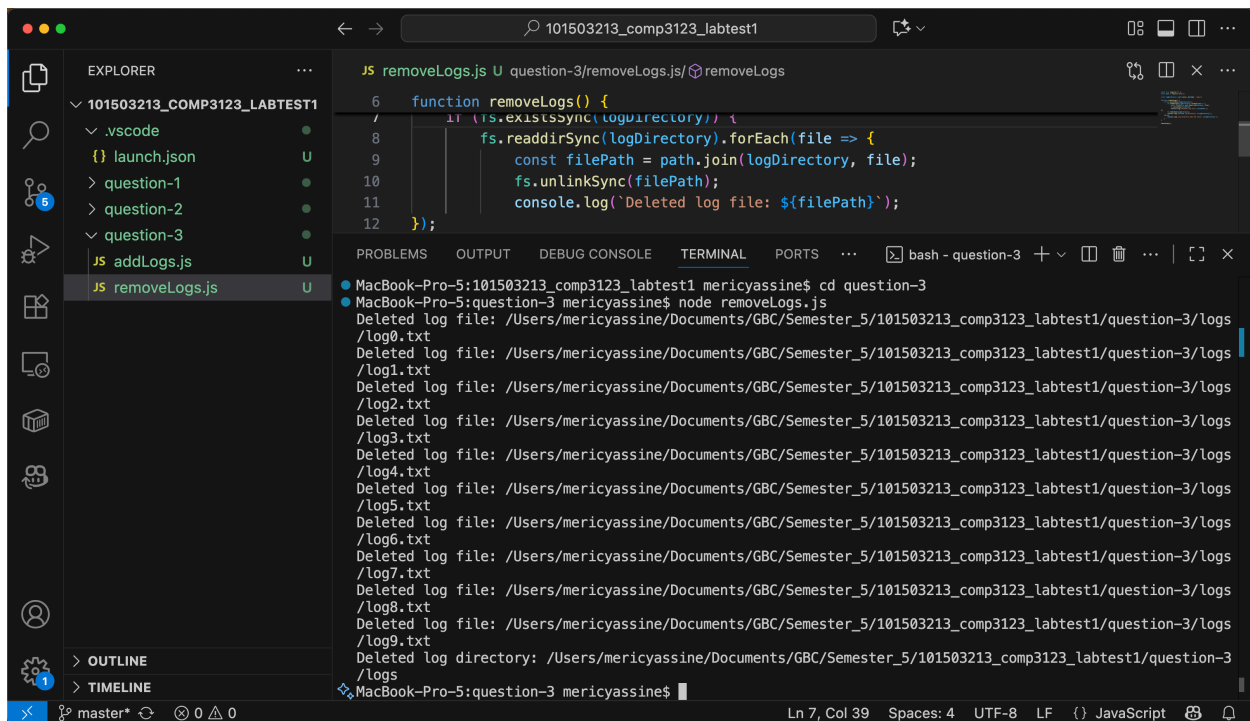
Question 3:



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of the project '101503213_COMP3123_LABTEST1'. The file 'addLogs.js' is selected. The main editor displays the code for 'addLogs.js', which uses the 'fs' module to create a log directory and log files. The terminal at the bottom shows the execution of 'node addLogs.js', which outputs the creation of a log directory and ten log files (log0.txt to log9.txt).

```
1 const fs = require('fs');
2 const path = require('path');
3
4 const logDirectory = path.join(__dirname, 'logs');
5
6 function addLogs() {
7   if (!fs.existsSync(logDirectory)) {
8     fs.mkdirSync(logDirectory);
9     console.log(`Created log directory: ${logDirectory}`);
10  }
11
12  process.chdir(logDirectory);
13
14  for (let i = 0; i < 10; i++) {
```

MacBook-Pro-5:101503213_comp3123_labtest1 mericyassine\$ cd question-3
MacBook-Pro-5:question-3 mericyassine\$ node addLogs.js
Created log directory: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs
Created log file: log0.txt
Created log file: log1.txt
Created log file: log2.txt
Created log file: log3.txt
Created log file: log4.txt
Created log file: log5.txt
Created log file: log6.txt
Created log file: log7.txt
Created log file: log8.txt
Created log file: log9.txt
MacBook-Pro-5:question-3 mericyassine\$



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of the project '101503213_COMP3123_LABTEST1'. The file 'removeLogs.js' is selected. The main editor displays the code for 'removeLogs.js', which uses the 'fs' module to remove log files. The terminal at the bottom shows the execution of 'node removeLogs.js', which outputs the deletion of the log directory and ten log files (log0.txt to log9.txt).

```
6 function removeLogs() {
7   if (fs.existsSync(logDirectory)) {
8     fs.readdirSync(logDirectory).forEach(file => {
9       const filePath = path.join(logDirectory, file);
10      fs.unlinkSync(filePath);
11      console.log(`Deleted log file: ${filePath}`);
12    });
13  }
14}
```

MacBook-Pro-5:101503213_comp3123_labtest1 mericyassine\$ cd question-3
MacBook-Pro-5:question-3 mericyassine\$ node removeLogs.js
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log0.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log1.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log2.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log3.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log4.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log5.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log6.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log7.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log8.txt
Deleted log file: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs/log9.txt
Deleted log directory: /Users/mericyassine/Documents/GBC/Semester_5/101503213_comp3123_labtest1/question-3/logs
MacBook-Pro-5:question-3 mericyassine\$