CM2010 Software Design and Development: Mid-term Coursework 1 submission [001] **Part 1 by Maryam Zaman**

Module Coupling Examples:

1. Content Coupling:

In the given JavaScript code, content coupling is observed in the fillBox and setPiece functions. The fillBox function relies on the internal details of the setPiece function, specifically its recursive nature and the condition to check whether the initial matrix slot is empty. Here is a source code showing content coupling:

‘Code source starts’

```
//When user clicks on a box
const fillBox = (e) => {
  // ...
  setPiece(5, colValue);
  // ...
};

const setPiece = (startCount, colValue) => {
  // ...
  if (initialMatrix[startCount][colValue] != 0) {
    startCount -= 1;
    setPiece(startCount, colValue);
  }
  // ...
};
```

‘Code source ends’

The fillBox function depends on the specific logic and conditions within setPiece, indicating content coupling. Changes in the implementation of setPiece could affect the behavior of fillBox.

2. **Data Coupling:**

In the given code, the winCheck function is data coupled with the functions checkAdjacentRowValues, checkAdjacentColumnValues, and checkAdjacentDiagonalValues, because winCheck communicates with these functions by passing data (row and column) through parameters. The functions are independent of each other and communicate through this data Here is a source code showing data coupling:

‘Code source starts’

```
//Win check logic
const winCheck = (row, column) => {
  //if any of the functions return true we return true
  return checkAdjacentRowValues(row)
    ? true
    : checkAdjacentColumnValues(column)
    ? true
    : checkAdjacentDiagonalValues(row, column)
    ? true
    : false;
};
```

‘Code source ends’

In the context of the game, these functions are checking if the current move (defined by the row and column) meets certain win conditions. If any of the functions return true, the winCheck function also returns true, indicating a win. If none of the functions return true, the winCheck function returns false, indicating that the current move does not result in a win. This shows how the winCheck function uses the data (row and column) to interact with the other functions, demonstrating data coupling.

Module Cohesion Examples:

1. Functional Cohesion:

In the given code, matrixCreator function is an example of functional cohesion. Code source is given below:

‘Code source starts’

```
//Create Matrix
const matrixCreator = () => {
 for (let innerArray in initialMatrix) {
  let outerDiv = document.createElement("div");
  outerDiv.classList.add("grid-row");
  outerDiv.setAttribute("data-value", innerArray);
  for (let j in initialMatrix[innerArray]) {
   //Set all matrix values to 0
   initialMatrix[innerArray][j] = [0];
   let innerDiv = document.createElement("div");
   innerDiv.classList.add("grid-box");
   innerDiv.setAttribute("data-value", j);
   innerDiv.addEventListener("click", (e) => {
    fillBox(e);
   });
   outerDiv.appendChild(innerDiv);
  }
  container.appendChild(outerDiv);
 }
};
```

‘Code source ends’

In matrixCreator function, all the operations are related to a single task: creating a matrix and initializing it with zeros. The function iterates over initialMatrix, creates div elements for each cell, sets an event listener for each cell, and appends these cells to the container. All these operations contribute to the single purpose of the function, which is to create and initialize a matrix. Therefore, this function exhibits functional cohesion.

2. Sequential Cohesion:

The startGame function could be an example of sequential cohesion. Code source is below:

‘Code source starts’

```
//Initialise game
window.onload = startGame = async () => {
  //Between 1 and 2
  currentPlayer = generateRandomNumber(1, 3);
  container.innerHTML = "";
  await matrixCreator();
  playerTurn.innerHTML = `Player <span>${currentPlayer}'s</span> turn`;
};
```

‘Code source ends’

In startGame function, the operations are related and are executed sequentially:

1. currentPlayer = generateRandomNumber(1, 3); - This line generates a random number between 1 and 3 and assigns it to currentPlayer. This could determine which player goes first.
2. container.innerHTML = ""; - This line clears the game board (represented by the container element). This prepares the game board for a new game.
3. await matrixCreator(); - This line creates a new game board. It uses the matrixCreator function, creates and initializes a matrix (or game board).
4. playerTurn.innerHTML = `Player <span>${currentPlayer}'s</span> turn`;- This line updates the display to indicate which player's turn it is. It uses thecurrentPlayer` value that was generated in the first step.

Each of these operations is a necessary step in the process of starting a new game, and each step follows logically from the previous one. The output of one step serves as the input to the next. Therefore, this function exhibits sequential cohesion.

Word limit: 492 (Excluding the module title and my name at the top and the code source along with the words 'code starts and ends')