I. Conceptual Coverage (40 points)

- **Summary of Work**:
  I used a matrix to store the state of the game board; I used loops to run the game loop and the title screen, and to convert the board from 1's and 0's to an easier to read format when displaying the board to the player; I used a switch statement to determine whether to display the title screen or the game; if statements for determining which side of the board to remove the matches from, and if the row was empty and could be moved to the bottom; outputs are displayed to the terminal and the information necessary to provide an input is displayed to the user when they are asked for inputs; I also split drawing the board into a seperate function.
- **Evaluation**:
  I believe I used all programming concepts taught to us in the creation of the game.
- **Evidence**:
  Matrix:

```
board = [
0, 0, 0, 1, 0, 0, 0;
0, 0, 1, 1, 1, 0, 0;
0, 1, 1, 1, 1, 1, 0;
1, 1, 1, 1, 1, 1, 1;
];
```

  Loop:

```
matchCount = 0;
  for r = 1:height(board)
      for c = 1:length(board)
          if board(r, c) == 1
              fprintf('I');
              matchCount = matchCount+1;
          else
              fprintf(' ');
          end
      end
      if r == 1
          row1MatchCount = matchCount;
      end
      fprintf('\n');
  end
```

  Switch Statement:

```
switch state
```

```
    case "title"
        ...
    case "game"
        ...
end
```

If Statement:

```
if player == 1
    board(1, find(head(board,1), toRemove, "first")) = 0;
else
    board(1, find(head(board,1), toRemove, "last")) = 0;
end
if ismember(1, head(board,1)) == false
    for i = 1:height(board)-1
        board(i, :) = board(i+1, :);
    end
    board(height(board), :) = [];
end
```

Output/Input:

```
fprintf("Choose option: (Play, Quit, Define (What is Nim?)) \n")
choice = input("> ", "s");
```

Function:

```
function [matchCount, row1MatchCount] = drawBoard(board)
    % For loop
    matchCount = 0;
    for r = 1:height(board)
        for c = 1:length(board)
            if board(r, c) == 1
                fprintf('I');
                matchCount = matchCount+1;
            else
                fprintf(' ');
            end
        end
        if r == 1
            row1MatchCount = matchCount;
        end
        fprintf('\n');
    end
end
```

- **Self-assigned Score**: 40 / 40

2. Value-add (20 points)

- **Summary of Enhancements**:
  Additions I made over a more basic version of the game include having a title-screen with ascii art; having a option to have the game Nim defined to the user in-case they are unfamiliar; I went through the extra effort of making the board easier to read instead of just chucking `disp(board);` in the code and calling it a day; and removing the matches from different sides of the board depending on which player removes them.
- **Personal Contribution**:
  All code was written by myself without the use of AI tools. The ASCII art was generated using a free online tool: https://patorjk.com/software/taag/
- **Code Quality**:
  `main.m` was 102 lines long & `drawBoard.m` was 18 lines long in their final iterations totalling 120 lines of code. Code is split into sections via the switch statement in the main file along with `%%` comments that seem to resemble headings within MatLab. Pieces of code had comments saying what the code would do though without explanation, more comments could've been used.
- **Self-assigned Score**: 16 / 20

---

3. Code Testing (15 points)

- **Testing Approach**:
  Code was tested during development by running the file and seeing the output and testing different inputs manually, a better approach could've been to split the file into many functions like what was done with the `drawBoard()` function then writing a script that would feed values to the functions and check the output however the approach taken is sufficient for a small project like this one.
  Structured test files were created, containing a set of inputs to test every possible function present at each stage of development.
- **Evaluation (rubric-based)**:
  - Excellent (15 pts): Careful, staged testing of all major components.
  - Substantial (11.25 pts): Multiple stages tested with some gaps.
  - Pass (7.5 pts): Basic evidence of testing for at least one stage.
  - Below Expectations (3.75 pts): Minimal/basic testing only.
  - Absent (0 pts): No testing evidence.
- **Self-assigned Score**: 15 / 15

---

4. Incremental Development (15 points)

- **Development Strategy**:
  All stages of development were recorded using Git with a commit at each milestone.
- **Evaluation (rubric-based)**:
  - Excellent (15 pts): Well-commented intermediate files showing small, testable increments.
  - Substantial (11.25 pts): Commented intermediate files with some increments.
  - Pass (7.5 pts): Some signs of staged development.
  - Below Expectations (3.75 pts): Minimal development trail.
  - Absent (0 pts): No development path evident.
- **Self-assigned Score**: 15 / 15

---

5. Code Style & Comments (10 points)

- **Evaluation (rubric-based)**:
  - Excellent (10 pts): All good style elements present across all versions.
  - Substantial (7.5 pts): Nearly all elements, only minor lapses.
  - Pass (5 pts): Some good style, inconsistently applied.
  - Below Expectations (2.5 pts): Partial adherence to code style.
  - Absent (0 pts): Inconsistent or missing good coding style.
- **Examples**:
  Comments were usually fairly descriptive in what something does:
  `% Switch statement to differentiate between title screen and game`
  `% Display info to user and get input for matches to remove`
  All variable names are single word or CamelCase except for the iterators of for loops within `drawBoard.m`, those use single letter short hand, row → r & column → c. Longer variable names could've been used that a more descriptive, e.g. `state` → `gameState` or `toRemove` → `matchesToRemove`.
- **Self-assigned Score**: 7.5 / 10

Score: 93.5 / 100