Algorithm

1. **Initialization** 1.1. Define constants:
    - `MAX_BOARDS_PER_TICKET = 10`
    - `LOTTO_COST_PER_BOARD = 5`
    - `LOTTO_PLUS_COST_PER_BOARD = 2.5`
    - `TOTAL_NUMBERS = 52`
2. **Initialize Variables**
    - `userType` (admin or user)
    - `userSelections` (array to store selected numbers)
    - `totalTickets` (array to store tickets)
    - `drawResults` (array to store draw results)
    - `winningTickets` (array to store winning tickets)
3. **Application Initialization**
    - Call `initializeApp()`
        - Display user interface
        - Show option to switch between admin and user
        - Show number selection interface
4. **Switch User Type**
    - Function `switchUserType(type)`
        - Set `userType` to `type`
        - If `type` is 'admin', call `displayAdminInterface()`
        - Else, call `displayUserInterface()`
5. **Display User Interface**
    - Function `displayUserInterface()`
        - Display number selection interface
        - Allow users to select 6 numbers from 1 to 52
        - Display cost calculation
        - Option to enter Lotto Plus 1 and Lotto Plus 2
        - Input number of boards
6. **Handle Number Selection**
    - Function `selectNumbers(selectedNumbers)`
        - Set `userSelections` to `selectedNumbers`
        - Call `calculateCost()`
7. **Calculate Cost**
    - Function `calculateCost()`
        - Calculate `baseCost` = `userSelections.length * LOTTO_COST_PER_BOARD`

- Calculate `lottoPlus1Cost` = (`userSelections` includes 'Lotto Plus 1') ? `userSelections.length * LOTTO_PLUS_COST_PER_BOARD` : 0
- Calculate `lottoPlus2Cost` = (`userSelections` includes 'Lotto Plus 2') ? `userSelections.length * LOTTO_PLUS_COST_PER_BOARD` : 0
- Calculate `totalCost` = `baseCost + lottoPlus1Cost + lottoPlus2Cost`
- Call `displayTotalCost(totalCost)`

8. **Generate Boards**
    - Function `generateBoards(numberOfBoards)`
        - Initialize `boards` as an empty array
        - For each board (0 to `numberOfBoards`):
            - Call `promptUserForNumbers()`
            - Append the returned numbers to `boards`
        - Call `assignBoardsToTickets(boards)`
9. **Prompt User for Numbers**
    - Function `promptUserForNumbers()`
        - Display interface for number selection
        - Return selected numbers
10. **Assign Boards to Tickets**
    - Function `assignBoardsToTickets(boards)`
        - Initialize `ticket` as an empty array
        - For each board in `boards`:
            - If `ticket.length` < `MAX_BOARDS_PER_TICKET`, append board to `ticket`
            - Else, append `ticket` to `totalTickets`, and set `ticket` to a new array with the current board
        - If `ticket.length` > 0, append `ticket` to `totalTickets`
        - Call `saveTickets()`
11. **Save Tickets**
    - Function `saveTickets()`
        - Save `totalTickets` to `localStorage` as 'tickets'
12. **Display Admin Interface**
    - Function `displayAdminInterface()`
        - Display draw simulation option
        - Display data for winning tickets
13. **Simulate Draw**
    - Function `simulateDraw()`
        - Set `drawResults` to the result of `generateRandomNumbers()`

        - Call `checkWinningTickets()`
        - Call `saveDrawResults()`

14. **Generate Random Numbers**
    - Function `generateRandomNumbers()`
        - Generate 6 unique random numbers from 1 to 52
        - Return the generated numbers

15. **Check Winning Tickets**
    - Function `checkWinningTickets()`
        - For each ticket in `totalTickets`:
            - For each board in the ticket:
                - Set `matches` to the result of `checkMatches(board, drawResults)`
                - If `matches` >= 3, append `{ board, matches }` to `winningTickets`
        - Call `alertWinners()`

16. **Check Matches**
    - Function `checkMatches(board, results)`
        - Compare `board` numbers with `results`
        - Return the number of matches

17. **Save Draw Results**
    - Function `saveDrawResults()`
        - Save `drawResults` to `localStorage` as 'drawResults'
        - Save `winningTickets` to `localStorage` as 'winningTickets'

18. **Alert Winners**
    - Function `alertWinners()`
        - Notify users if they won
        - Notify admin of winning tickets

19. **Display Total Cost**
    - Function `displayTotalCost(cost)`
        - Update UI to show the total cost

20. **Handle User Notifications**
    - Function `handleUserNotifications()`
        - Check if user has winning tickets and notify them

21. **Initialize the Application**
    - Call `initializeApp()`

This algorithm outlines the steps and functions needed to implement the Lotto Simulator, covering initialization, user interaction, ticket generation, draw simulation, and notifications.

Pseudocode for Task 3 - Lotto Simulator (Part One)

```
// Define constants
const MAX_BOARDS_PER_TICKET = 10;
const LOTTO_COST_PER_BOARD = 5;
const LOTTO_PLUS_COST_PER_BOARD = 2.5;
const TOTAL_NUMBERS = 52;

// Initialize variables
let userType; // 'admin' or 'user'
let userSelections = [];
let totalTickets = [];
let drawResults = [];
let winningTickets = [];

// Function to initialize the application
function initializeApp() {
// Display user interface
// Show option to switch between admin and user
// Show number selection interface
}

// Function to handle user type switch
function switchUserType(type) {
userType = type;
if (type === "admin") {
displayAdminInterface();
} else {
displayUserInterface();
}
}

// Function to display user interface
function displayUserInterface() {
// Display number selection interface
// Allow users to select 6 numbers from 1 to 52
// Display cost calculation
// Option to enter Lotto Plus 1 and Lotto Plus 2
// Input number of boards
}

// Function to handle number selection by users
function selectNumbers(selectedNumbers) {
userSelections = selectedNumbers;
calculateCost();
}
```

```
// Function to calculate cost based on selected options
function calculateCost() {
let baseCost = userSelections.length * LOTTO_COST_PER_BOARD;
let lottoPlus1Cost = userSelections.includes("Lotto Plus 1")
? userSelections.length * LOTTO_PLUS_COST_PER_BOARD
: 0;
let lottoPlus2Cost = userSelections.includes("Lotto Plus 2")
? userSelections.length * LOTTO_PLUS_COST_PER_BOARD
: 0;
let totalCost = baseCost + lottoPlus1Cost + lottoPlus2Cost;
displayTotalCost(totalCost);
}

// Function to handle board generation
function generateBoards(numberOfBoards) {
let boards = [];
for (let i = 0; i < numberOfBoards; i++) {
let board = promptUserForNumbers();
boards.push(board);
}
assignBoardsToTickets(boards);
}

// Function to prompt user for numbers
function promptUserForNumbers() {
// Display interface for number selection
// Return selected numbers
}

// Function to assign boards to tickets
function assignBoardsToTickets(boards) {
let ticket = [];
for (let i = 0; i < boards.length; i++) {
if (ticket.length < MAX_BOARDS_PER_TICKET) {
ticket.push(boards[i]);
} else {
totalTickets.push(ticket);
ticket = [boards[i]];
}
}
if (ticket.length > 0) {
totalTickets.push(ticket);
}
saveTickets();
}
```

```
// Function to save tickets to localStorage
function saveTickets() {
localStorage.setItem("tickets", JSON.stringify(totalTickets));
}

// Function to display admin interface
function displayAdminInterface() {
// Display draw simulation option
// Display data for winning tickets
}

// Function to simulate a draw
function simulateDraw() {
drawResults = generateRandomNumbers();
checkWinningTickets();
saveDrawResults();
}

// Function to generate random numbers for draw
function generateRandomNumbers() {
// Generate 6 random numbers from 1 to 52
// Return the generated numbers
}

// Function to check winning tickets
function checkWinningTickets() {
totalTickets.forEach((ticket) => {
ticket.forEach((board) => {
let matches = checkMatches(board, drawResults);
if (matches >= 3) {
winningTickets.push({ board, matches });
}
});
});
alertWinners();
}

// Function to check matches between board and draw results
function checkMatches(board, results) {
// Compare board numbers with draw results
// Return number of matches
}

// Function to save draw results
function saveDrawResults() {
localStorage.setItem("drawResults", JSON.stringify(drawResults));
localStorage.setItem("winningTickets", JSON.stringify(winningTickets));
```

```
}

// Function to alert winners
function alertWinners() {
// Notify users if they won
// Notify admin of winning tickets
}

// Function to display total cost
function displayTotalCost(cost) {
// Update UI to show the total cost
}

// Function to handle user notifications
function handleUserNotifications() {
// Check if user has winning tickets and notify them
}

// Initialize the application
initializeApp();
```