```
`timescale 1ns / 1ps
module lab4_2(
    input [1:0] mode, // Modes: 01 for Entrance, 00 for Exit, 10 for
Search, 11 for List
    input [5:0] userID, // Format: [AreaID(2 bits)][UserID(4 bits)]
    input CLK,
    output reg [1:0] selectedAreaId, // Shows user-selected area
    output reg [5:0] numberOfInsideUser, // Number of users inside
    output reg [3:0] listOutput, // Lists the IDs of students in the
area (4 bits for UserID)
    output reg AlreadyInside, // Indicates if a student is inside
    output reg NotInside // Indicates if a student is not inside
);

    // Define the memory for seating records
    reg [15:0] seating[3:0]; // 4 areas with 16 seats each, using
bit vector to represent each seat

    // Internal variables
    reg [3:0] userPosition; // Extracted user position from userID
    reg [15:0] areaSeats; // Seats in the selected area
    integer i; // Loop variable

    // Additional variables for list mode
    reg [3:0] listIndex; // Index to track the position in List Mode
    reg found; // Flag to indicate if a seat was found

    // Initialization
    initial begin
        // Initialize all seating to zero
        for (i = 0; i < 4; i = i + 1) begin
            seating[i] = 16'b0;
        end
        AlreadyInside = 0;
        NotInside = 0;
        listIndex = 0;
    end

    always @(posedge CLK) begin
        // Extract the area ID and user position from userID
        selectedAreaId = userID[5:4];
        userPosition = userID[3:0];
        areaSeats = seating[selectedAreaId];

        // Reset flags
        AlreadyInside = 0;
        NotInside = 0;

        case (mode)
            2'b01: begin // Student Entrance Mode
                if (areaSeats[userPosition] == 1'b1) begin
                    AlreadyInside = 1;
                end else begin
                    seating[selectedAreaId][userPosition] = 1;
```

```
                end
            end

            2'b00: begin // Student Exit Mode
                if (areaSeats[userPosition] == 1'b0) begin
                    NotInside = 1;
                end else begin
                    seating[selectedAreaId][userPosition] = 0;
                end
            end

            2'b10: begin // Search Mode
                if (areaSeats[userPosition] == 1'b1) begin
                    AlreadyInside = 1;
                end else begin
                    NotInside = 1;
                end
            end

            2'b11: begin // List Mode
                // Reset the list index to start from the beginning
each time list mode is entered
                listIndex = 0;
                // Listing logic will be handled in a separate
always block
            end
        endcase

        // Update number of users inside
        numberOfInsideUser = 0;
        for (i = 0; i < 16; i = i + 1) begin
            if (seating[selectedAreaId][i] == 1'b1) begin
                numberOfInsideUser = numberOfInsideUser + 1;
            end
        end
    end

    // Combinational logic for List Mode
    always @* begin
        if (mode == 2'b11) begin
            // Find the next occupied seat in increasing order
            listOutput = 4'b0000; // Default value when no user is
found
            found = 0; // Reset the found flag
            for (i = listIndex; i < 16 && !found; i = i + 1) begin
                if (seating[selectedAreaId][i] == 1'b1) begin
                    listOutput = i;
                    listIndex = i + 1;
                    found = 1; // Set the found flag
                end
            end
            // Reset listIndex if we reach the end without finding
any more occupied seats
            if (!found) begin
```

```
                listIndex = 0;
            end
        end else begin
            listIndex = 0; // Reset listIndex when not in list mode
        end
    end

endmodule
```