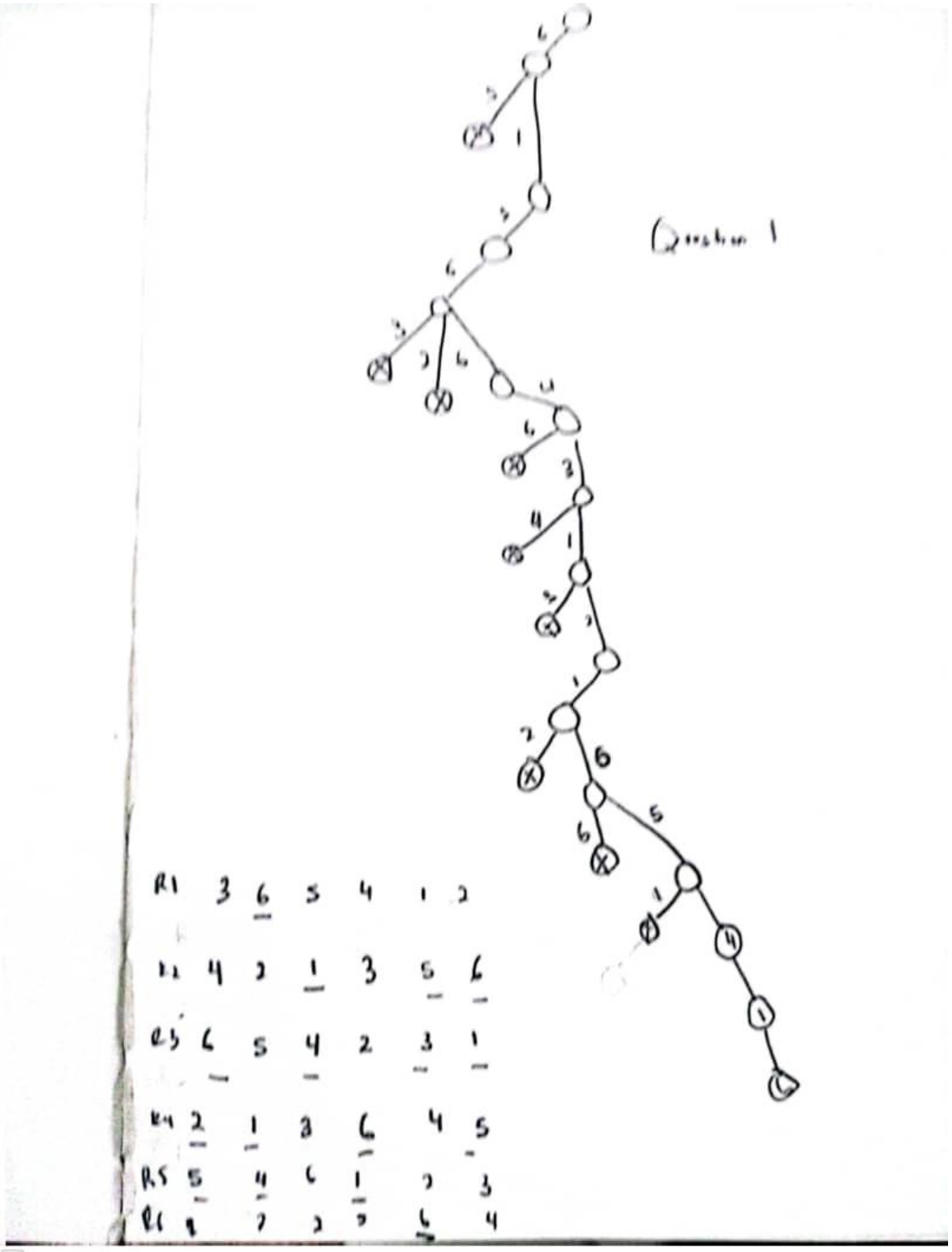
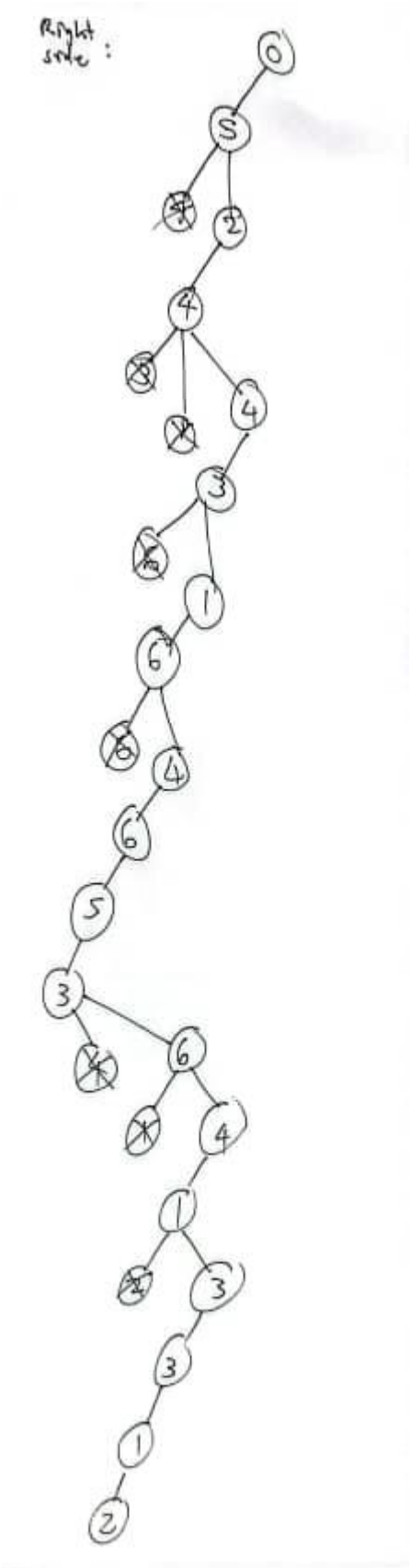
Question 1





Question 2

```
#include <stdio.h>
```

```
#include <stdbool.h>

#define SIZE 6

// Function to print the Sudoku board
void print_board(int board[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("%d ", board[i][j]);
        }
        printf("\n");
    }
}

// Function to check if a number can be placed in the given position
bool is_valid(int board[SIZE][SIZE], int row, int col, int num) {
    // Check if the number is not present in the row and column
    for (int i = 0; i < SIZE; i++) {
        if (board[row][i] == num || board[i][col] == num) {
            return false;
        }
    }

    // Check if the number is not present in the 2x3 box
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[row - row % 2 + i][col - col % 3 + j] == num) {
                return false;
            }
        }
    }
```

```
    return true;
}

// Function to find the next empty cell
bool find_empty_location(int board[SIZE][SIZE], int *row, int *col) {
    for (*row = 0; *row < SIZE; (*row)++) {
        for (*col = 0; *col < SIZE; (*col)++) {
            if (board[*row][*col] == 0) {
                return true; // Found an empty cell
            }
        }
    }
    return false; // No empty cell found
}

// Recursive function to solve the Sudoku puzzle
bool solve_sudoku(int board[SIZE][SIZE]) {
    int row, col;

    // If there is no empty cell, the puzzle is solved
    if (!find_empty_location(board, &row, &col)) {
        return true;
    }

    // Try placing numbers 1 through 6 in the empty cell
    for (int num = 1; num <= 6; num++) {
        if (is_valid(board, row, col, num)) {
            // Place the number if it's valid
            board[row][col] = num;
```

```
            // Recursively solve the rest of the puzzle
            if (solve_sudoku(board)) {
                return true;
            }

            // If placing the number leads to a contradiction, backtrack
            board[row][col] = 0;
        }
    }

    // If no number can be placed, backtrack to the previous decision
    return false;
}

int main() {
    // Define the Sudoku puzzle
    int sudoku_board[SIZE][SIZE] = {
        {3, 0, 5, 4, 1, 2},
        {4, 2, 0, 3, 0, 0},
        {0, 5, 0, 2, 0, 0},
        {0, 0, 3, 0, 4, 0},
        {0, 0, 6, 0, 2, 3},
        {1, 3, 2, 5, 0, 4}
    };

    // Solve the Sudoku puzzle using recursive backtracking
    if (solve_sudoku(sudoku_board)) {
        printf("Sudoku solved successfully:\n");
        print_board(sudoku_board);
    } else {
        printf("No solution exists.\n");
```

```
    }

    return 0;
}
```