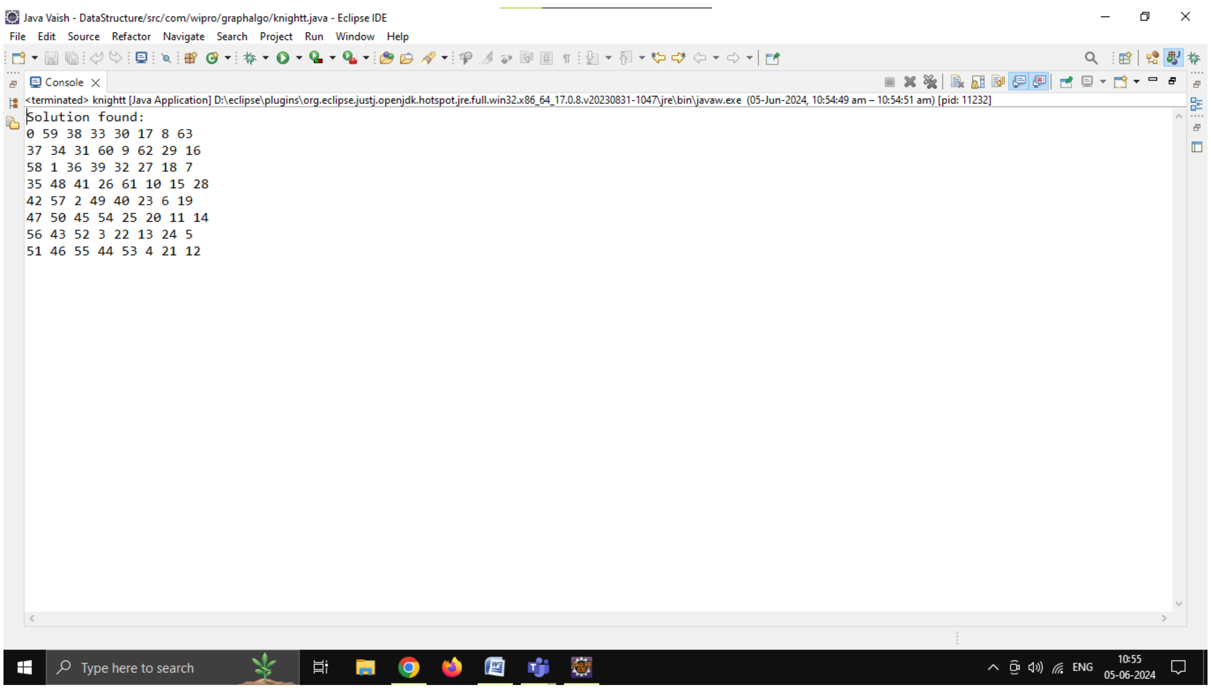
Day 16 and 17:

Task 1: The Knight's Tour Problem

Create a function bool SolveKnights Tour(int[,] board, int moveX, int moveY, int moveCount, int[] xMove, int[] yMove) that attempts to solve the Knight's Tour problem using backtracking. The function should return true if a solution exists and false otherwise. The board represents the chessboard, moveX and movey are the current coordinates of the knight, moveCount is the current move count, and xMove[], yMove[] are the possible next moves for the knight. Fill the chessboard such that the knight visits every square exactly once. Keep the chessboard size to 8x8.

Answer:

```
package com.wipro.graphalgo;
public class knightt {

    static int N = 8;
    static boolean isValid(int x, int y, int[][] board) {
        return (x >= 0 && y >= 0 && x < N && y < N && board[x][y] ==
-1);
    }

    static void printSolution(int[][] board) {
        for (int x = 0; x < N; x++) {
            for (int y = 0; y < N; y++)
                System.out.print(board[x][y] + " ");
            System.out.println();
        }
    }

    static boolean solveKTUtil(int x, int y, int moveCount, int[][]
board, int[] xMove, int[] yMove) {
        int k, nextX, nextY;
        if (moveCount == N * N)
            return true;

        for (k = 0; k < 8; k++) {
            nextX = x + xMove[k];
            nextY = y + yMove[k];
            if (isValid(nextX, nextY, board)) {
                board[nextX][nextY] = moveCount;
                if (solveKTUtil(nextX, nextY, moveCount + 1, board,
xMove, yMove))
                    return true;
                else
                    board[nextX][nextY] = -1;
            }
        }
        return false;
    }
    static boolean solveKnightsTour(int[][] board, int startX, int
startY) {
        int[] xMove = { 2, 1, -1, -2, -2, -1, 1, 2 };
        int[] yMove = { 1, 2, 2, 1, -1, -2, -2, -1 };

        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++)
                board[i][j] = -1;

        board[startX][startY] = 0;

        if (!solveKTUtil(startX, startY, 1, board, xMove, yMove)) {
            System.out.println("Solution does not exist");
            return false;
        } else {
            System.out.println("Solution found:");
            printSolution(board);
            return true;
        }
    }

    public static void main(String[] args) {
        int[][] board = new int[N][N];
        int startX = 0;
        int startY = 0;

        solveKnightsTour(board, startX, startY);
    }
}
```



```
<terminated> knightt [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe (05-Jun-2024, 10:54:49 am – 10:54:51 am) [pid: 11232]
Solution found:
0 59 38 33 30 17 8 63
37 34 31 60 9 62 29 16
58 1 36 39 32 27 18 7
35 48 41 26 61 10 15 28
42 57 2 49 40 23 6 19
47 50 45 54 25 20 11 14
56 43 52 3 22 13 24 5
51 46 55 44 53 4 21 12
```

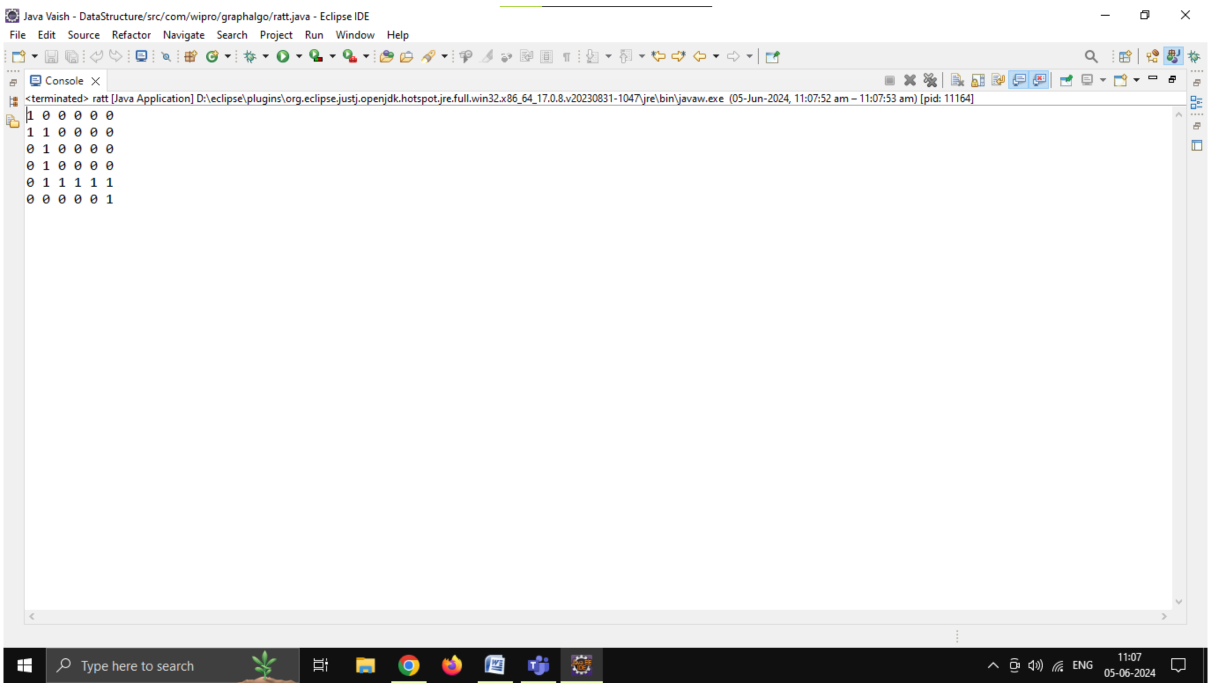
Task 2: Rat in a Maze

mplement a function bool SolveMaze(int[,] maze) that uses backtracking to find a path from the top left corner to the bottom right corner of a maze. The maze is represented by a 2D array where 1s are paths and Os are walls. Find a rat's path through the maze. The maze size is 6x6.

Answer:

```
package com.wipro.graphalgo;
public class ratt {

    static final int N = 6;

    static boolean isSafe(int[][] maze, int x, int y) {
        return (x >= 0 && x < N && y >= 0 && y < N && maze[x][y] ==
1);
    }

    static boolean solveMaze(int[][] maze, int x, int y, int[][]
sol) {
        if (x == N - 1 && y == N - 1) {
            sol[x][y] = 1;
            return true;
        }
```

```java
        if (isSafe(maze, x, y)) {
            sol[x][y] = 1;

            if (solveMaze(maze, x + 1, y, sol))
                return true;

            if (solveMaze(maze, x, y + 1, sol))
                return true;

            sol[x][y] = 0;
            return false;
        }

        return false;
    }
    static void printSolution(int[][] sol) {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++)
                System.out.print(sol[i][j] + " ");
            System.out.println();
        }
    }

    public static void main(String[] args) {
        int[][] maze = {
            {1, 0, 0, 0, 0, 0},
            {1, 1, 1, 1, 0, 1},
            {0, 1, 0, 1, 1, 1},
            {0, 1, 0, 0, 0, 1},
            {1, 1, 1, 1, 1, 1},
            {0, 0, 0, 0, 0, 1}
        };

        int[][] sol = new int[N][N];
        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++)
                sol[i][j] = 0;

        if (solveMaze(maze, 0, 0, sol))
            printSolution(sol);
        else
            System.out.println("No solution exists");
    }
}
```


Java Vaish - DataStructure/src/com/wipro/graphalgo/ratt.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Console
Type here to search
ENG
11:07
05-06-2024

```
<terminated> ratt [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (05-Jun-2024, 11:07:52 am – 11:07:53 am) [pid: 11164]
1 0 0 0 0 0
1 1 0 0 0 0
0 1 0 0 0 0
0 1 0 0 0 0
0 1 1 1 1 1
0 0 0 0 0 1
```

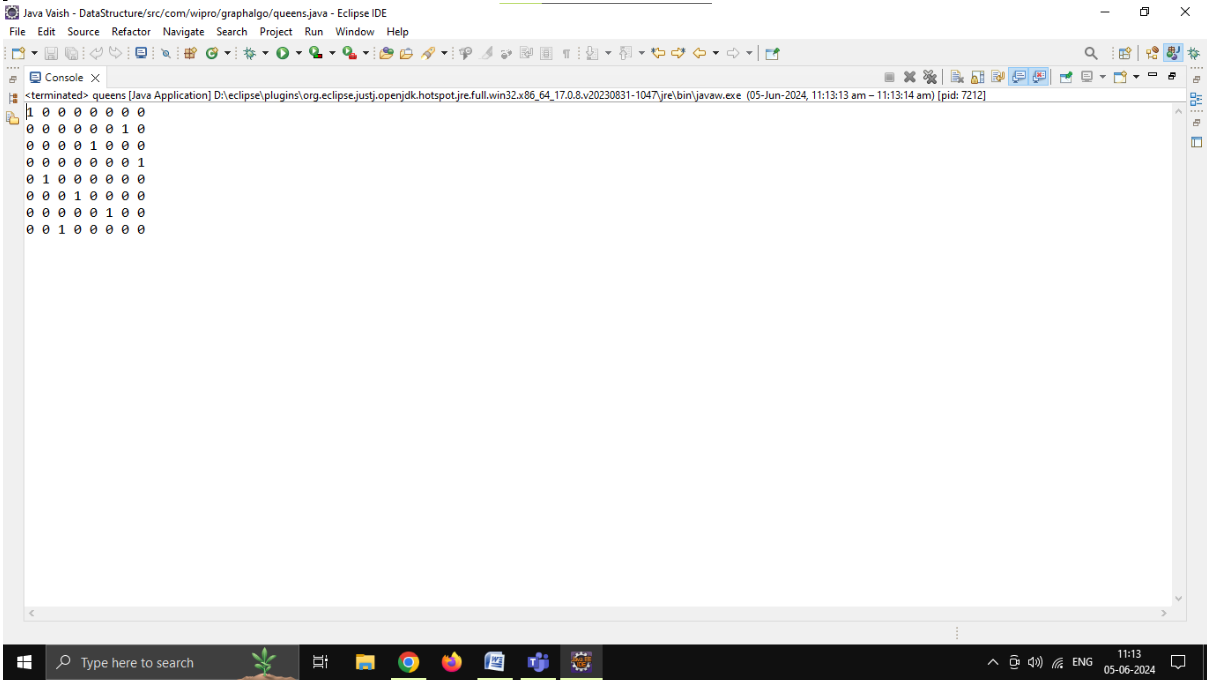
Task 3: N Queen Problem

Write a function bool SolveNQueen(int[,] board, int col) in C# that places N queens on an N x N chessboard so that no two queens attack each other using backtracking. Place N queens on the board such that no two queens can attack each other. Use a standard 8x8 chessboard.

Answer:

```java
package com.wipro.graphalgo;
public class queens {
    static int N = 8;
    static boolean isSafe(int[][] board, int row, int col) {
        int i, j;

        for (i = 0; i < col; i++)
            if (board[row][i] == 1)
                return false;
        for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
            if (board[i][j] == 1)
                return false;
        for (i = row, j = col; j >= 0 && i < N; i++, j--)
            if (board[i][j] == 1)
                return false;

        return true;
}

static boolean solveNQueensUtil(int[][] board, int col) {

    if (col >= N)
        return true;

    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {

            board[i][col] = 1;

            if (solveNQueensUtil(board, col + 1))
                return true;

            board[i][col] = 0;
        }
    }

    return false;
}

static boolean solveNQueens(int[][] board) {
    if (!solveNQueensUtil(board, 0)) {
        System.out.println("Solution does not exist");
        return false;
    }

    printSolution(board);
    return true;
}

static void printSolution(int[][] board) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            System.out.print(board[i][j] + " ");
        System.out.println();
    }
}

public static void main(String[] args) {
    int[][] board = new int[N][N];

    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            board[i][j] = 0;

    solveNQueens(board);
}
```

```
}
```



Java Vaish - DataStructure/src/com/wipro/graphalgo/queens.java - Eclipse IDE

```
<terminated> queens [Java Application] D:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (05-Jun-2024, 11:13:13 am – 11:13:14 am) [pid: 7212]
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```