

## Group Members:

- Zeeshan Ali (01-134212-197)
- Shayan Hassan Abbasi (01-134212-167)
- Muhammad Sheraz Ajmal (01-134212-130)
- Syed Shahzaib Naseeb Shah (01-134212-178)

## DAA PROJECT Part-B

---

### Problem Statement:

This game is played on the grid. The F counter represents a farmer; the R counter represents a rooster. The farmer and the rooster move alternately until the rooster is captured. On each move, each of them can move to a neighboring point on the grid: up, down, left, or right. A capture occurs when the farmer can move on a point occupied by the rooster.

### Objective:

The objective of the Rooster Chase problem is to implement the game logic and simulate the movements of the farmer and the rooster on the grid, checking for the capture conditions and determining the final outcome of the game.

### Code:

```
#include <iostream>
using namespace std;

// Function To Display The Grid
void displayGrid(char grid[][8]) {
    const int gridSize = 8;
    for (int i = 0; i < gridSize; i++) {
        for (int j = 0; j < gridSize; j++) {
            cout << grid[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
}
```

```

// Function For One-Player Mode
void playOnePlayer() {
    system("cls");
    cout << "One Player Mode\n";

    int iF, jF;
    int iR, jR;

    // Initial Position Of Farmer At Top Left
    iF = 1;
    jF = 1;

    // Initial Position Of Rooster At Bottom Right
    iR = 8;
    jR = 8;

    // Size Of Grid
    const int gridSize = 8;

    // Moves Counting Of Farmer & Rooster
    int TotalMovesF = 0;
    int TotalMovesR = 0;

    // Create & Display Grid
    char grid[gridSize][gridSize];
    for (int i = 0; i < gridSize; i++) {
        for (int j = 0; j < gridSize; j++) {
            grid[i][j] = '-';
        }
    }
    grid[iF - 1][jF - 1] = 'F'; // Mark Farmer's Position
    grid[iR - 1][jR - 1] = 'R'; // Mark Rooster's Position
    displayGrid(grid);

    // Loop For Maximum Of 14 Moves For Farmer
    for (int move = 1; move <= 14; move++) {
        // Player's Move
        if (iF == iR && jF == jR) {
            cout << "Rooster captured!\n";
            cout << "Moves Taken By Farmer : " << TotalMovesF;
            cout << "\nMoves Taken By Rooster : " << TotalMovesR;
            char playAgain;
            cout << "\nDo you want to play again? (y/n): ";
            cin >> playAgain;
            if (playAgain == 'y' || playAgain == 'Y') {
                return; // Return To Main Menu
            }
            else {
                cout << "Thank you for playing!\n";
                exit(0); // Exit Program
            }
        }
    }
}

```

```

}

cout << "Move the farmer (u, d, l, r): ";
char direction;
cin >> direction;

switch (direction) {
case 'u':
    if (iF > 1) {
        iF--;
        TotalMovesF++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
case 'd':
    if (iF < gridSize) {
        iF++;
        TotalMovesF++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
case 'l':
    if (jF > 1) {
        jF--;
        TotalMovesF++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
case 'r':
    if (jF < gridSize) {
        jF++;
        TotalMovesF++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
}

```

```

default:
    cout << "Invalid move. Try again.\n";
    move--;
    continue; // Repeat the loop
}

// Rooster's Move
if (iR > iF) {
    iR--; // Move up
    TotalMovesR++;
}
else if (jR > jF) {
    jR--; // Move left
    TotalMovesR++;
}

// Check if the farmer and rooster are adjacent
if (abs(iF - iR) <= 1 && abs(jF - jR) <= 1) {
    grid[iF - 1][jF - 1] = 'F'; // Mark final farmer's position
    grid[iR - 1][jR - 1] = 'R'; // Mark final rooster's position
    displayGrid(grid);
    cout << "Rooster captured!\n";
    cout << "Moves Taken By Farmer : " << TotalMovesF;
    cout << "\nMoves Taken By Rooster : " << TotalMovesR;
    char playAgain;
    cout << "\nDo you want to play again? (y/n): ";
    cin >> playAgain;
    if (playAgain == 'y' || playAgain == 'Y') {
        return; // Return to the main menu
    }
    else {
        cout << "Thank you for playing!\n";
        exit(0); // Exit the program
    }
}

// Display the updated grid
grid[iF - 1][jF - 1] = 'F';
grid[iR - 1][jR - 1] = 'R';
displayGrid(grid);
}

// Check if the rooster is not captured after all moves
cout << "Rooster is not captured\n";
cout << "Press Enter Key To Return To Main Screen";
cin.ignore(); // Ignore any previous newline characters in the buffer
cin.get();
}

// Function For Two-Player Mode
void playTwoPlayer() {

```

```

system("cls");
cout << "Two Player Mode\n";
int iF, jF;
int iR, jR;

// Initial Position Of Farmer At Top Left
iF = 1;
jF = 1;

// Initial Position Of Farmer At Bottom Right
iR = 8;
jR = 8;

// Size Of Grid
const int gridSize = 8;

// Moves Counting Of Farmer & Rooster
int TotalMovesR = 0;
int TotalMovesF = 0;

// Create and display the grid
char grid[gridSize][gridSize];
for (int i = 0; i < gridSize; i++) {
    for (int j = 0; j < gridSize; j++) {
        grid[i][j] = '-';
    }
}
grid[iF - 1][jF - 1] = 'F'; // Mark farmer's position
grid[iR - 1][jR - 1] = 'R'; // Mark rooster's position
displayGrid(grid);

// Loop for a maximum of 28 moves (14 moves for each player)
for (int move = 1; move <= 28; move++) {
    // Farmer's move
    if (move % 2 != 0) {
        cout << "Move the farmer (u, d, l, r): ";
        char direction;
        cin >> direction;

        switch (direction) {
            case 'u':
                if (iF > 1) {
                    iF--;
                    TotalMovesF++;
                }
                else {
                    cout << "You Tried To Move Outside Grid. Try Again\n";
                    move--;
                    continue; // Repeat the loop
                }
            break;

```

```

case 'd':
    if (iF < gridSize) {
        iF++;
        TotalMovesF++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
case 'l':
    if (jF > 1) {
        jF--;
        TotalMovesF++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
case 'r':
    if (jF < gridSize) {
        jF++;
        TotalMovesF++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
default:
    cout << "Invalid move. Try again.\n";
    move--;
    continue; // Repeat the loop
}
}
else {
    // Rooster's move
    cout << "Move the rooster (u, d, l, r): ";
    char direction;
    cin >> direction;

    switch (direction) {
case 'u':
        if (iR > 1) {
            iR--;
            TotalMovesR++;
        }

```

```

else {
    cout << "You Tried To Move Outside Grid. Try Again\n";
    move--;
    continue; // Repeat the loop
}
break;
case 'd':
    if (iR < gridSize) {
        iR++;
        TotalMovesR++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
case 'l':
    if (jR > 1) {
        jR--;
        TotalMovesR++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
case 'r':
    if (jR < gridSize) {
        jR++;
        TotalMovesR++;
    }
    else {
        cout << "You Tried To Move Outside Grid. Try Again\n";
        move--;
        continue; // Repeat the loop
    }
    break;
default:
    cout << "Invalid move. Try again.\n";
    move--;
    continue; // Repeat the loop
}
}

```

```

// Check if the farmer and rooster are adjacent
if (abs(iF - iR) <= 1 && abs(jF - jR) <= 1) {
    grid[iF - 1][jF - 1] = 'F'; // Mark final farmer's position
    grid[iR - 1][jR - 1] = 'R'; // Mark final rooster's position
    displayGrid(grid);
}

```

```

        cout << "Rooster captured!\n";
        cout << "Moves Taken By Farmer : " << TotalMovesF;
        cout << "\nMoves Taken By Rooster : " << TotalMovesR;
        char playAgain;
        cout << "\nDo you want to play again? (y/n): ";
        cin >> playAgain;
        if (playAgain == 'y' || playAgain == 'Y') {
            return; // Return to the main menu
        }
        else {
            cout << "Thank you for playing!\n";
            exit(0); // Exit the program
        }
    }

    // Display the updated grid
    grid[iF - 1][jF - 1] = 'F';
    grid[iR - 1][jR - 1] = 'R';
    displayGrid(grid);
}

// Check if the rooster is not captured after all moves
cout << "Rooster is not captured\n";
cout << "Press Enter Key To Return To Main Screen";
cin.ignore(); // Ignore any previous newline characters in the buffer
cin.get();
}

void Rules() {
    system("cls");
    cout << "***** RULES OF GAME *****\n\n";
    cout << "ONE PLAYER MODE:\n";
    cout << "1. You Can Only Move Farmer. Rooster Will Move Automatically\n";
    cout << "2. You Can Move Up, Down, Left & Right\n";
    cout << "3. You Will Get Only 14 Moves To Capture The Rooster\n";
    cout << "4. If You Failed To Catch Rooster In 14 Moves, Game Over\n";
    cout << "\nTWO PLAYER MODE:\n";
    cout << "1. One Player Will Move The Farmer And Other Player Will Move The Rooster\n";
    cout << "2. Both Farmer And Rooster Can Move Up, Down, Left & Right\n";
    cout << "3. Each Player Will Get Only 14 Moves\n";
    cout << "4. Farmer Will Try To Catch Rooster And Rooster Will Try To Escape Farmer\n";
    cout << "5. If Farmer Catch The Rooster, Farmer Wins Else Rooster Wins\n\n";
    cout << "NOTE: Neither Farmer Nor Rooster Can Move Outside The Grid!!\n\n";
    cout << "Press Enter Key To Return To Main Screen";
    cin.ignore(); // Ignore any previous newline characters in the buffer
    cin.get();
}

// Main function
int main() {
    while (true) {

```



```
system("cls");
cout << "Welcome to the Rooster Catching Game!\n";
cout << "Choose one choice:\n";
cout << "1. One Player\n";
cout << "2. Two Player\n";
cout << "3. How To Play?\n\n";

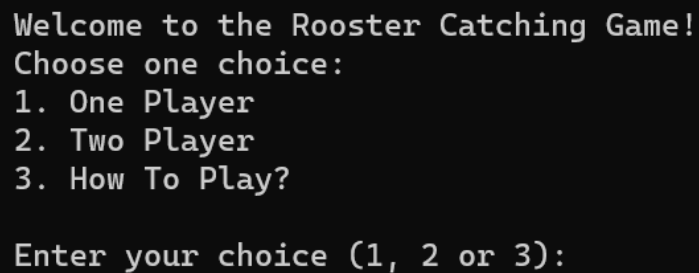
int choice;
cout << "Enter your choice (1, 2 or 3): ";
cin >> choice;

switch (choice) {
case 1:
    playOnePlayer();
    break;
case 2:
    playTwoPlayer();
    break;
case 3:
    Rules();
    break;
default:
    cout << "Invalid choice. Try again.\n";
    break;
}
}

return 0;
}
```

## **Output Sample:**

### 1. Main Screen



```
Welcome to the Rooster Catching Game!
Choose one choice:
1. One Player
2. Two Player
3. How To Play?

Enter your choice (1, 2 or 3):
```

## 2. One-Player Mode

One Player Mode

```
F - - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - - R
```

Move the farmer (u, d, l, r): d

```
F - - - - -  
F - - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - - R  
- - - - - R
```

Move the farmer (u, d, l, r): r

```
F - - - - -  
F F - - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - - R  
- - - - - R  
- - - - - R
```

Move the farmer (u, d, l, r): |

Move the farmer (u, d, l, r): r

```
F - - - - -  
F F - - - - -  
- F - - - - -  
- F F F R R R R  
- - - - - R  
- - - - - R  
- - - - - R  
- - - - - R
```

Rooster captured!

Moves Taken By Farmer : 7

Moves Taken By Rooster : 7

Do you want to play again? (y/n):

### 3. Two-Player Mode

```
Two Player Mode
```

```
F - - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - - R
```

```
Move the farmer (u, d, l, r): d
```

```
F - - - - -  
F - - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - - R
```

```
Move the rooster (u, d, l, r): u
```

```
F - - - - -  
F - - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - - R  
- - - - - R
```

```
Move the farmer (u, d, l, r): r
```

```
F - - - - -  
F - - - - -  
F - - - - -  
F F F F F R R R  
- - - - - R  
- - - - - R  
- - - - - R  
- - - - - R
```

```
Rooster captured!
```

```
Moves Taken By Farmer : 7
```

```
Moves Taken By Rooster : 6
```

```
Do you want to play again? (y/n):
```

#### 4. Rules

\*\*\*\*\* RULES OF GAME \*\*\*\*\*

ONE PLAYER MODE:

1. You Can Only Move Farmer. Rooster Will Move Automatically
2. You Can Move Up, Down, Left & Right
3. You Will Get Only 14 Moves To Capture The Rooster
4. If You Failed To Catch Rooster In 14 Moves, Game Over

TWO PLAYER MODE:

1. One Player Will Move The Farmer And Other Player Will Move The Rooster
2. Both Farmer And Rooster Can Move Up, Down, Left & Right
3. Each Player Will Get Only 14 Moves
4. Farmer Will Try To Catch Rooster And Rooster Will Try To Escape Farmer
5. If Farmer Catch The Rooster, Farmer Wins Else Rooster Wins

NOTE: Neither Farmer Nor Rooster Can Move Outside The Grid!!

Press Enter Key To Return To Main Screen|

#### 5. Invalid Moves

One Player Mode

```
F - - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - -  
- - - - - R
```

```
Move the farmer (u, d, l, r): u  
You Tried To Move Outside Grid. Try Again  
Move the farmer (u, d, l, r): q  
Invalid move. Try again.  
Move the farmer (u, d, l, r): |
```

#### **Conclusion:**

This problem is solved by using Greedy algorithm strategy. The functionality of this algorithm is to show whether the Farmer chased down the Rooster or not within 14 Moves. All the necessary conditions and requirements were kept in mind while designing the algorithm.