ST MARY UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

COURSE: INTRODUCTION COMLUTER PROGRAMING

GROUP NUMBER 4

TITLE: TIC-TAC-TOE GAME

GROUP MEMBERS	ID NUMBER
1) Abel Baysa	RCD/0229/2017
2) Adoniyas Melaku	RCD/0232/2017
3) Ahmed yesuf	RCD/0234/2017
4) Dawit ketsela.	RCD/0248/2017
5) Kirubel mekonnen	RCD/0265/2017
6) Hryakos Gorfu	ECD/1262/2017
7) Haileyesus Abera	
8) Eman Ali.	RCD/0253/2017

OVERVIEW

This C++ program implements a simple Tic-Tac-Toe game for two players using basic programming constructs.

FUNCTIONALITY

Game Board

- Uses a 3x3 grid represented as a one-dimensional array
- Initially displays numbers 1 through 9 to indicate position selection
- Updates and redraws the board after each valid move

Player System

- Player 1 is assigned X as their marker
- Player 2 is assigned O as their marker
- Players alternate turns automatically
- Clear indication shows which player's turn it is

Game Logic

Win Detection:

- Checks all three rows for matching markers
- Checks all three columns for matching markers
- Checks both diagonal lines for matching markers
- Immediately declares winner when detected

Draw Detection:

- Checks if all board positions are filled
- Verifies no winning condition exists

- Declares a draw when appropriate

Input Handling

- Accepts position numbers from 1 to 9
- Validates input range (must be 1-9)
- Prevents overwriting occupied positions
- Provides clear error messages for invalid moves

Game Flow

- Uses a do-while loop for main game cycle
- Displays updated board after each move
- Automatically switches players after valid moves
- Ends game when win or draw condition met
- Shows final result before exiting

Technical Implementation

- Uses basic control structures (if, while, for)
- Implements switch-case for input validation
- Employs arrays for board storage
- Includes string for player display
- Utilizes break for flow control
- Maintains clean code structure

The game provides a complete Tic Tac Toe experience while adhering to specified programming constraints and maintaining clear, straightforward gameplay.

CODE SNIPPET

```
#include <iostream>
#include <string>
Using namespace std;

Int main() {
  Char board[9] = {‘1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’,’9’};
  Int player = 1;
  Int choice;
  Bool game_over = false;
  String player_names[2] = {“Player 1 (X)”, “Player 2 (O)”};

  Do {
    // Display the board
    Cout << “\n\nTic Tac Toe\n\n”;
    Cout << player_names[0] << “ – “ << player_names[1] << “\n\n”;
    For (int I = 0; I < 9; I += 3) {
      Cout << “    |    |    \n”;
      Cout << “  “ << board[i] << “  |  “ << board[i+1] << “  |  “ << board[i+2] << “  \n”;
      If (I < 6) cout << “_____|_____|_____\n”;
      Else cout << “    |    |    \n”;
    }
```

```
// Check win conditions
For (int I = 0; I < 3; i++) {
  // Check rows
  If (board[i*3] == board[i*3+1] && board[i*3+1] == board[i*3+2]) {
    Game_over = true;
    Break;
  }
  // Check columns
  If (board[i] == board[i+3] && board[i+3] == board[i+6]) {
    Game_over = true;
    Break;
  }
}
// Check diagonals
If ((board[0] == board[4] && board[4] == board[8]) ||
  (board[2] == board[4] && board[4] == board[6])) {
  Game_over = true;
}

If (game_over) {
  Cout << player_names[player-1] << " wins!\n";
  Break;
}

// Check for draw
```

```
Bool draw = true;
For (char cell : board) {
  If (cell != ‘X’ && cell != ‘O’) {
    Draw = false;
    Break;
  }
}
If (draw) {
  Cout << “Game Draw!\n”;
  Break;
}

// Player input
Cout << player_names[player-1] << “, enter a number (1-9): “;
Cin >> choice;

Switch(choice) {
  Case 1: case 2: case 3: case 4: case 5: case 6: case 7: case 8: case 9:
    If (board[choice-1] == ‘X’ || board[choice-1] == ‘O’) {
      Cout << “Position already taken!\n”;
      Continue;
    }
    Board[choice-1] = (player == 1) ? ‘X’ : ‘O’;
    Player = (player == 1) ? 2 : 1;
    Break;
  Default:
```

```
            Cout << “Invalid input! Please enter 1-9.\n”;
            Continue;
        }

    } while (true);

    Return 0;
}
```