# CCP PROJECT

# PF (CT-175)

**Group Name:**

Binary Breakers

**Group Members:**

Zuha Azhar (CT-25055)

Muhammad Ashar Hussain (CT-25087)

**Discipline:** BCIT

**Teachers:**

Sir Abdullah

Sir Furqan

## 1. Project Title

**CodeBreaker: A Logical and Engaging Number Guessing Game**

## 2. Project Description

CodeBreaker is a console-based logical number-guessing game developed in **C language**. The game challenges players to crack a secret numerical code within limited attempts, encouraging problem-solving, pattern recognition, and strategic deduction.

The program offers two primary modes:

- **Single Player Mode** — where a user plays against the computer.
- **VS Mode** — where two players compete, and the faster one wins.

Each mode uses **real-time timers**, **random number generation**, and **feedback-based gameplay**, providing both entertainment and a practical demonstration of programming fundamentals such as arrays, loops, conditionals, and time functions.

## 3. Project Methodology

### 3.1 Programming Tools & Environment

- **Language:** C
- **Compiler:** GCC
- **Version Control:** GitHub
- **IDE:** Dev C++ and VS Code
- **Libraries Used:**
    - stdio.h for input/output
    - stdlib.h for random number generation
    - time.h for timers and delays

### 3.2 Algorithm

1. **Menu Display:**
   The program first displays a mode selection menu.

- Option 1: Single Player Mode
- Option 2: Multiplayer (VS) Mode

2. **Random Code Generation:**
   Using rand() and srand(time(0)), random digits are generated to form a secret code.
3. **Gameplay Logic:**
   Players enter guesses separated by spaces. After each guess, the program provides feedback symbols:
   - # → Correct digit and correct position
   - ~ → Correct digit but wrong position
   - X → Digit not in the secret code
4. **Scoring & Timing:**
   - Single Player Mode: Score decreases with wrong attempts (starting from 115).
   - VS Mode: Timer comparison determines the winner.
5. **Termination Conditions:**
   - Player wins if all digits match before attempts run out.
   - Game ends when the player either wins or exhausts all attempts.

## 3.3 Objectives

- Develop a **fully functional console-based game** in C language.
- Implement a **feedback system** to guide the player's next move.
- Introduce **difficulty levels** (Easy, Medium, Hard).
- Add **score and timer tracking** for fairness and challenge.
- Strengthen **team collaboration and modular coding** skills.

## 3.4 Scope

- Randomized secret code generation.
- Input validation and result feedback.
- Scoring and limited attempts in Single Player Mode.

- Real-time timer-based competition in VS Mode.
- Variable difficulty levels.

**Scope Excludes:**
Graphical interface, database connectivity, or networking-based multiplayer.

## 3.5 Timeline

| **Week** | **Task** |
|---|---|
| **Week 7** | Proposal submission, GitHub repository setup, base code structure. |
| **Week 8** | Implementation of random generation, feedback system, and scoring logic. |
| **Week 9** | Input validation, timer integration, and VS mode development. |
| **Week 10** | Debugging, code optimization, and gameplay testing. |
| **Week 11–12** | Documentation, report writing, and final project demonstration. |

## 3.6 Expected Outcomes

- A fully functional **console-based game** showcasing logic and programming fundamentals.
- **Single Player and VS modes** with distinct gameplay mechanics.
- **Feedback and scoring systems** that simulate real-world logic processing.
- Modular, readable code with proper use of loops, conditionals, and functions.

## 3.7 Goals

- Successfully demonstrate problem-solving through programming logic.
- Build a complete, replayable game using only C fundamentals.
- Achieve structured code organization with clear modular design.
- Provide an enjoyable user experience while learning coding concepts.

## 4. Justification — Why It Is a Complex Computing Problem

This project qualifies as a **Complex Computing Problem (CCP)** because it involves:

- **Algorithm design:** Logical comparison between digits and positions.
- **Randomization:** Dynamic secret code generation for each gameplay session.
- **Timing mechanisms:** Real-time performance tracking using time_t.
- **User interaction and feedback:** Intelligent responses based on game logic.
- **Error handling and flow control:** Managing user input, scoring, and attempts.

Together, these elements demonstrate a deeper understanding of structured programming, algorithmic thinking, and problem-solving in C.

## 5. Industrialization / Future Potential

While CodeBreaker is currently a console-based project, its structure allows future expansion into:

- **Graphical User Interface (GUI)** using C++ or Python.
- **Mobile or Web versions** for broader accessibility.
- **Online Multiplayer Mode** using sockets or databases.
- **AI-based code generation or adaptive difficulty** for advanced gameplay.

This makes CodeBreaker a scalable foundation for interactive logic-based educational games.

## 6. Repository Link

🔗 https://github.com/zuhaazhar/Projects