C PROGRAMMING PROJECT

MINI GAME PROJECT

Submitted By:
Nandini Joshi
Sap ID: 590023287

Submitted To:
Mr. Rahul Prasad

BRANCH: School of Computer Science
BATCH NUMBER: 19
YEAR: 2025-26

1. Abstract

This project implements a **Mini Game Pack** in C containing multiple small games: **Guess the Number**, **Rock–Paper–Scissors**, and **Snake–Water–Gun**. The project demonstrates modular programming, use of functions, random number generation, menu-driven interfaces, input validation, and basic game logic. The application is intended for beginners to practice C programming concepts while playing simple interactive games.

2. Problem Definition

Design and implement a menu-driven C application that:

- Presents multiple mini-games inside one program.
- Implements each game as a separate function/module.
- Uses randomization where applicable (rand()).
- Provides smooth, safe navigation and handles invalid input.
- Is modular, readable, and easy to compile.

Objective: Build a simple, extensible Mini Game Pack demonstrating fundamentals of C programming.

3. Flowchart (Text Description)

1. Start
2. Display Welcome Screen
3. Seed RNG (`srand(time(NULL))`)
4. Show Main Menu
    - 1: Guess the Number
    - 2: Rock-Paper-Scissors
    - 3: Snake-Water-Gun
    - 4: Exit
5. Read user choice
6. Execute corresponding function
7. Return to menu after a game finishes.
8. Exit when user selects 4

4. Algorithm (High-level)

1. Start program
2. Call `WelcomeScreen()`
3. Seed RNG
4. Enter main menu loop
5. Read numeric choice safely (validate input)

6. On valid selection, call that game function
7. Game runs and returns to menu
8. Repeat until exit

5. System Design & Modules

- `include/game.h` — header file with function declarations.
- `src/main.c` — program entrypoint, seed RNG, call `WelcomeScreen()` and `MainMenu()`.
- `src/welcome.c` — prints welcome/cover message.
- `src/menu.c` — main menu loop, handles input validation, dispatches to game functions.
- `src/games.c` — implementations for `guessNumber()`, `rockPaperScissors()`, `snakeWaterGun()`, `Exit_sys()`.

6. Implementation (Full Source Code)

include/game.h

```
#ifndef GAME_H
#define GAME_H

void WelcomeScreen(void);
void MainMenu(void);
void guessNumber(void);
void rockPaperScissors(void);
void snakeWaterGun(void);
void Exit_sys(void);

#endif
```

src/main.c

```
#include "game.h"
#include <time.h>
#include <stdlib.h>

int main() {
    WelcomeScreen();
    srand(time(NULL));   // Seed random number generator
    MainMenu();
    return 0;
}
```

src/welcome.c

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

void WelcomeScreen() {
    printf("\n------------------------\n");
    printf("      MINI GAME PACK\n");
    printf("------------------------\n");
    printf("Press ENTER to continue...");
    getchar();
}
```

src/menu.c

```
#include <stdio.h>
#include <stdlib.h>
#include "game.h"
```

```
void MainMenu() {
    int ch;

    while (1) {
        printf("\n----- MINI GAME PACK -----\n");
        printf("1. Guess The Number\n");
        printf("2. Rock Paper Scissors\n");
        printf("3. Snake Water Gun\n");
        printf("4. Exit\n");
        printf("Enter choice: ");

        if (scanf("%d", &ch) != 1) {
            int x;
            while ((x = getchar()) != '\n' && x != EOF) {}
            continue;
        }

        if (ch == 1) guessNumber();
        else if (ch == 2) rockPaperScissors();
        else if (ch == 3) snakeWaterGun();
        else if (ch == 4) Exit_sys();
        else printf("Invalid choice\n");
    }
}
```

src/games.c

```
#include <stdio.h>
#include <stdlib.h>
#include "game.h"

void guessNumber() {
    int num = rand() % 100 + 1;
    int g, t = 0;

    printf("\nGuess a number between 1 to 100\n");

    do {
        printf("Enter guess: ");
        scanf("%d", &g);
        t++;

        if (g > num) printf("Too high\n");
        else if (g < num) printf("Too low\n");
        else printf("Correct! Attempts: %d\n", t);

    } while (g != num);

    getchar();
    getchar();
}
```

```
void rockPaperScissors() {
    int u, c;
    c = rand() % 3;

    printf("\n0=Rock 1=Paper 2=Scissors\n");
    printf("Enter choice: ");
    scanf("%d", &u);

    printf("Computer: %d\n", c);

    if (u == c) printf("Draw\n");
    else if ((u == 0 && c == 2) || (u == 1 && c == 0) || (u == 2 && c == 1)) printf("You win\n");
    else printf("Computer wins\n");

    getchar();
    getchar();
}

void snakeWaterGun() {
    int u, c;
    c = rand() % 3;

    printf("\n0=Snake 1=Water 2=Gun\n");
    printf("Enter choice: ");
    scanf("%d", &u);

    printf("Computer: %d\n", c);

    if (u == c) printf("Draw\n");
    else if ((u == 0 && c == 1) || (u == 1 && c == 2) || (u == 2 && c == 0)) printf("You win\n");
    else printf("Computer wins\n");

    getchar();
    getchar();
}

void Exit_sys() {
    printf("Thank you for playing\n");
    exit(0);
}
```

7. Testing & Results

Testing approach: Functional testing of each module with repeated runs and invalid inputs.

Test cases (examples):

- Menu selection: choose 1 → Guess the Number starts → Pass
- Invalid input handling: enter letter/symbol → program prompts and doesn't crash → Pass
- Guess the Number: multiple wrong guesses → hints displayed; attempts counted → Pass
- Rock–Paper–Scissors: 0/1/2 outcomes validated → Pass
- Snake–Water–Gun: outcomes validated → Pass

Sample input (for interactive testing) — not for direct piping but for human tester to follow:

1. Start program
2. Choose 1 → play Guess the Number
3. Choose 2 → play RPS
4. Choose 3 → play Snake-Water-Gun
5. Choose 4 → Exit

8. Conclusion

The Mini Game Pack demonstrates modular programming in C, use of randomness, menu-driven UI, and input handling. It is a good beginner project and easily extendable.

9. Future Enhancements

- Add more games (Tic-Tac-Toe, Hangman).
- Add high score and persistence (file-based).
- Add GUI (SDL or port to Pygame).
- Add difficulty levels and scoring.
- Add two-player local mode.