**PROJECT REPORT**

**DIGITAL-LOGIC DESIGN**

**BCS 2J**

****

**Title: SNAKE GAME IN C LANGUAGE**

**Group Members:**

**23K-0613 M. AHSAN BARI**

**23K-0535 MAHNOOR HUSSAIN**

**23K-0632 AHMED**

**Project’s Introduction:**The Snake Game is a classic arcade game where a player controls a snake that moves around the screen, eating food pellets to grow longer while avoiding collisions with walls and its own tail. The objective is to achieve the highest score possible without running into obstacles. This project aims to implement the Snake Game in the C programming language with additional features such as a score system, lives, loading screen, and saving scores to a file.

**Implementation:**

**Basic Snake Game:**

The core functionality of the game involves managing the snake's movement, detecting collisions, generating food pellets, and updating the game state accordingly. This is achieved using fundamental C programming concepts such as loops, conditionals, and data structures like arrays.

**Score System:**

A score system is implemented to keep track of the player's performance. Points are awarded for each food pellet eaten, and the score is displayed on the screen. The score increases as the snake consumes more food pellets.

**Lives:**

To add an element of challenge, the game includes a life system. The player starts with a set number of lives (usually three), and each collision with a wall or the snake's own body results in losing a life. When all lives are lost, the game ends.

**Loading Screen:**

A loading screen is displayed when the game is launched, providing a visually appealing introduction to the game. This screen may include game title, instructions, and perhaps a simple animation to engage the player.

**Scores Saved in a File:**

After each game session, the player's score is saved to a file. This allows the player to track their progress over multiple sessions and provides a form of persistence between gameplay sessions.

**Code Structure:**

The code is organized into modular functions, each responsible for a specific aspect of the game. This modular approach enhances readability, maintainability, and reusability of the codebase. Additionally, comments are included throughout the code to explain the purpose of each function and significant sections.

**Code Explanation:**

let's break down the code and explain each part, including the header files used:

1. Header Files:

- `stdio.h`: Standard Input Output header file, used for input and output functions like `printf` and `scanf`.

- `time.h`: Header file for time functions like `time()` which is used to get current time.

- `stdlib.h`: Standard Library header file, used for functions like `rand()` for generating random numbers, `srand()` for seeding the random number generator.

- `conio.h`: Console Input Output header file, provides functions like `getch()` for reading a character from the console without echoing it.

- `ctype.h`: Character handling functions like `toupper()` for converting characters to uppercase.

- `windows.h`: Windows-specific functions like `SetConsoleCursorPosition()` for setting the cursor position in the console.

- `process.h`: Process Control header file, provides functions like `exit()` for terminating the program.

2. Macros:

- `UP`, `DOWN`, `LEFT`, `RIGHT`: Constants representing arrow key codes.

3. Global Variables:

- `length`: Length of the snake.

- `bend no`: Number of bends in the snake.

- `Len`: Current length of the snake body.

- `key`: Variable to store user input.

- `life`: Number of lives for the player.

- `head`, `bend[]`, `food`, `body[]`: Structures representing coordinates of the snake's head, bends, food, and body segments.

4. Function Prototypes:

- Prototypes for functions defined later in the code.

5. Main Function:

- Initializes variables and starts the game loop.

6. Functions:

- `Move()`: Controls the movement of the snake.

- `gotoxy()`, `GotoXY()`: Functions for setting the cursor position on the console.

- `load()`: Displays a loading animation.

- `Down()`, `Left()`, `Right()`, `Up()`: Functions for moving the snake in different directions.

- `Delay()`: Delays execution for a certain period.

- `ExitGame()`: Handles game over conditions.

- `Food()`: Generates food for the snake.

- `Bend()`: Draws bends in the snake's body.

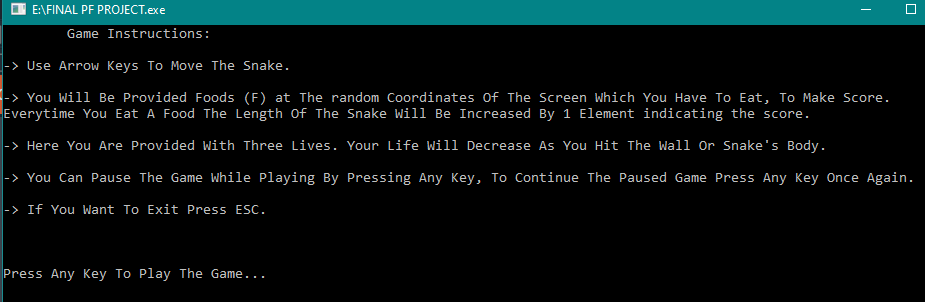
- `Border()`: Draws the game border.

- `Print()`: Prints the welcome message and game instructions.

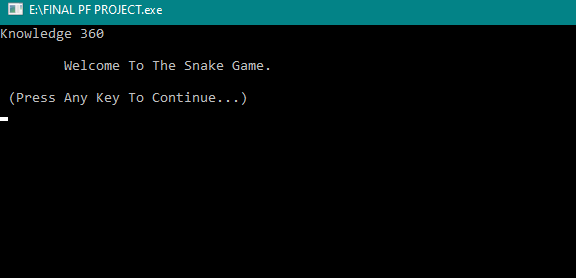
- `record()`: Records player information and score.

- `Score()`, `Scoreonly()`: Calculates and displays the player's score.

**Output Snippets:**

****

****

****

**Conclusion:**

In conclusion, the Snake Game implemented in C language successfully incorporates essential features such as a score system, lives, loading screen, and saving scores to a file. The project demonstrates proficiency in C programming and provides an entertaining gaming experience for users. With further enhancements and refinements, the game could be even more engaging and enjoyable for players.