DEPARTMENT OF INFORMATION ENGINEERING TECHNOLOGY

SUPERIOR UNIVERSITY LAHORE

SNAKE GAME

PROJECT PROPOSAL BY:

RIMSHA	SU91-BIETM-F23-037
AREEBA	SU91-BIETM-F23-038
HARIS ALI	SU91-BIETM-F23-057

SUBJECT: DATA STRUCTURE AND ALGORITHM

PROJECT ADVISOR: MISS RANIYA

Table of Contents

Abstract:.....3
Introduction: 3
Problem Statement: 3
Objective:.....3
Methodology: 4
Conclusion:.....5
References: 5

Abstract:

This project is a console-based **Snake Game** developed in **C++**, utilizing dynamic memory allocation through linked lists to manage the snake's body segments. The snake moves on a 20x40 grid, consuming randomly placed food items. Each piece of food increases the snake's length and score. The game ends when the snake collides with the grid boundaries or itself. The player's score is saved in a high-score file, allowing comparison with previous best scores.

Introduction:

The **Snake Game** is a classic game that gained popularity in the 1970s and later became a staple in early mobile gaming. In this implementation, the player controls a snake that moves continuously in the specified direction and grows by consuming food represented by an asterisk (*). The goal is to maximize the score without crashing into the walls or the snakes own body. This project is a demonstration of various **C++ programming concepts**, including data structures, memory management, file handling, and real-time input processing.

Problem Statement:

Develop a simple Snake Game using C++ where the player controls a snake to collect food represented by * on a 2D grid. The snake grows in length as it consumes food and the game ends if the snake collides with itself or the boundaries of the board. The game should maintain high scores and display them at the start.

Objective:

The objective of this project is:

- To implement a fully functional Snake Game using C++.
- To demonstrate the use of dynamic memory allocation and linked lists. Utilizing linked lists to represent the snake's body.
- To apply file handling for maintaining high scores.
- To provide real-time input handling and screen updates.

- Implement logic to detect collisions with walls and the snake's own body, food consumption, and score tracking.
- Clearing and updating the console to simulate game.

Methodology:

Game Initialization:

- Initialize the random number generator to create random food locations.
- Prompt the player for their name and initialize the snake's position.
- Display the current high score from a file named highscore.txt.

Snake Class:

- **Attributes:** head, tail, length.
- **Methods:**
  - SNAKE_HEAD(int x, int y): Adds a new head node to the snake.
  - SNAKE_Movement(char direction, bool grow): Moves the snake in the specified direction and grows if it eats food.
  - Display_Snake(char board[20][40]): Renders the snake on the board.

Game Loop:

- Continuously clear and update the console screen.
- Capture user input for snake direction using _kbhit() and _getch().
- Check for food consumption and self-collision.
- End the game if the snake hits the boundary or itself.

Food Generation:

- generateFood (int Board_W, int Board_H, Snake& snake): Randomly places food on the board, avoiding the snake's current position.

Screen and Board Handling:

- Display_Board (char board[20][40]): Displays the current state of the board with the snake and food.
- clearScreen (): Clears the console using Windows API functions for smooth animation.

High Score Handling:

- gameOver (int score, const string& playerName): Saves the score and displays a "Game Over" message.
- DisplayHighestScore (): Reads and displays the highest score from highscore.txt.

Conclusion:

The Snake Game demonstrates basic game mechanics using dynamic memory allocation and linked lists. The project highlights file handling for score storage and uses Windows API for screen manipulation. Future enhancements could include adding levels, obstacles, or a pause feature.

References:

- C++ Standard Library for input/output operations.
- Windows API documentation for screen manipulation.
- Tutorials on using conio.h and linked lists for game development.