Course Code: CSE115 — Group 10 — Section 04
Abdur Rafiu Bhuiyan (ID: 2521470042), Tarannum Islam (ID: 2522707642),
Nishithini Nishuti Roy (ID: 2523711042), Imad Yashfi Khan (ID: 2524318642)

***Abstract*—This paper presents the design and partial implementation of a configurable word search puzzle generator in the C programming language. The proposed system allows user-defined grid dimensions and dynamic word insertion, supporting multiple orientations (horizontal, vertical, and diagonal). This report focuses on the first half of the implementation, covering grid initialization, input handling, and direction modeling.**

***Index Terms*—Word Search, C Programming, Puzzle Generator, Grid Algorithms, Educational Software**

I. Introduction

Word search puzzles are a widely enjoyed form of entertainment and educational activity. Algorithmically generating such puzzles presents challenges in spatial planning, randomness, and data integrity.

Our C implementation enables grid and word list customization, directional placement, and efficient grid validation. The system emphasizes portability and clarity using basic C structures and functions.

II. System Overview

The system is divided into:

- **User Input Handler:** Accepts grid size and words.
- **Direction Model:** Encodes 8 possible directions.
- **Validation Routine:** Verifies word fits.
- **Grid Model:** 2D character matrix.

III. Grid and Direction Modeling

A. Grid Structure

We define:

```
char grid[MAX_GRID][
    MAX_GRID];
char solution[MAX_GRID][
    MAX_GRID];
```

Initialized as:

```
for (int i = 0; i <
    GRID_SIZE; i++)
  for (int j = 0; j <
      GRID_SIZE; j++) {
   grid[i][j] = '.';
   solution[i][j] = 0;
}
```

B. Direction Encoding

To support 8 movement directions:

```
typedef struct {
   int dx, dy;
} Direction;

Direction directions[] =
    {
   {0, 1}, {1, 0}, {0,
       -1}, {-1, 0},
   {1, 1}, {1, -1}, {-1,
       1}, {-1, -1}
};
```

IV. Input and Initialization

User inputs grid size and list of words. The grid is filled with `'.'` indicating empty space, and later words or random letters.

Design and Implementation of a Dynamic Word Search Puzzle Generator in C

V. Word Placement Strategy

Function to validate placement:

```
int is_valid_position(int
     x, int y, int dx,
    int dy, const char*
    word) {
  int len = strlen(word);
  for (int i = 0; i < len
      ; i++) {
    int nx = x + dx * i;
    int ny = y + dy * i;
    if (nx < 0 || ny < 0
        || nx >= GRID_SIZE
         || ny >=
        GRID_SIZE)
      return 0;
    if (grid[nx][ny] !=
        '.' && grid[nx][ny
        ] != word[i])
      return 0;
  }
  return 1;
}
```

VI. Complexity Analysis

Let n be grid size, w number of words, and l average word length.

$$T(n) = O(w \cdot n^2 \cdot 8 \cdot l) = O(8wn^2l)$$

Each cell is checked across 8 directions for each character in the word list.

VII. Conclusion

This paper introduced a lightweight word search puzzle generator in C. The grid model and direction encoding support flexible configuration. Future development includes GUI features, backward search, and saving solutions to file.