ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 223E

DATA STRUCTURES
HOMEWORK REPORT

HOMEWORK NO : 1

HOMEWORK DATE : 14.07.2025

820220336 : UYGAR GÖK

SUMMER 2025

Contents

1 INTRODUCTION

In software projects that simulate or model biological systems, game dynamics, or artificial environments, dynamic data structures such as linked lists play a key role in managing flexible and changing entities. One such application is the simulation of worms that exist on a coordinate grid. The attack worms function is designed as a core mechanic in such a system, where the user or an agent attempts to "attack" a specific point in space to hit and damage these worm structures.

This function navigates through a list of worms, each represented as a doubly linked list of segments, and identifies if any segment exists at the targeted (x, y) coordinates. If a segment is found, its adjacent nodes are conditionally removed, simulating damage to the worm, and a subsequent function may reorganize or split the worm body into new entities. This operation involves both coordinate-based logic and precise memory management, demonstrating common patterns in system level C programming.

The purpose of this report is to provide a detailed breakdown of the structure, logic and operation of the attack worms function, including the data models it interacts with, the algorithm it follows, and potential improvements or risks associated with its implementation.

2 METHODS

The `attack_worms` function is designed to simulate a targeted attack on a worm segment within a 2D grid-based environment. Each worm is represented as a doubly linked list, where each node corresponds to a segment at a specific `(x, y)` coordinate.
When the function is called, it iterates through all active worms and checks if any segment matches the targeted coordinates. If a match is found, the function simulates damage by removing adjacent segments—,one before and one after,—and updates the list structure accordingly.
In cases where a segment is hit, the function calls `split_and_replace()` to evaluate and restructure the remaining worm parts, potentially splitting them into new worm entities. This approach ensures both logical consistency of the simulation and efficient memory handling through proper deallocation of removed segments. If no segment is found at the given position, the function simply reports a missed attack.

3 RESULTS

The implementation of the `attack_worms` function successfully achieves its intended purpose of simulating targeted damage within a dynamic worm structure. When a segment at a specific coordinate is hit, the function ensures accurate modification of the worm's linked list by conditionally removing neighboring segments and maintaining list integrity. Memory is properly managed through explicit deallocation of removed nodes, and the invocation of the `split_and_replace` function allows the simulation to dynamically update and restructure worm entities in response to attacks.

Test scenarios demonstrate that the function reliably identifies hit and miss cases, provides clear output messages to inform the user, and preserves the stability of the remaining data structure.

Overall, the function contributes to a robust simulation mechanism and serves as a key part of the interaction logic within a grid-based environment.

4 CONCLUSION [10 points]

In conclusion, the `attack_worms` function provides a clear and functional approach to managing targeted interactions within a dynamic, list-based simulation. Its correct handling of memory, structural updates, and modular design highlights effective programming practices. While some edge cases could be addressed more robustly, the function as implemented performs reliably within its intended scope and serves as a solid foundation for further development in grid-based simulation systems.

REFERENCES