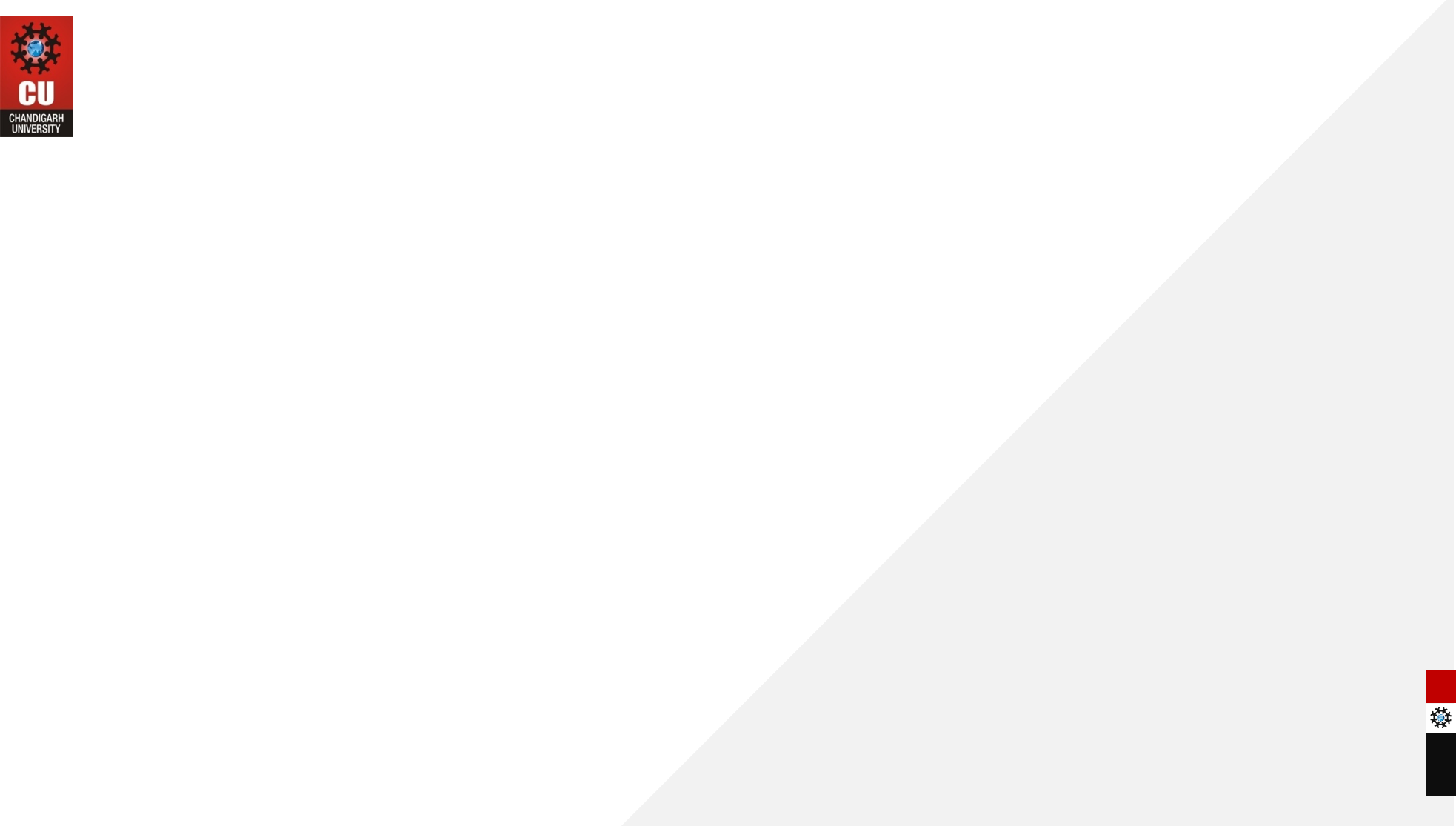
Food Delivery System using Data Structures in C

(Graphs, Queues, Linked Lists)

Presented by: [MANIK BABBAR] UID=24BCA10423

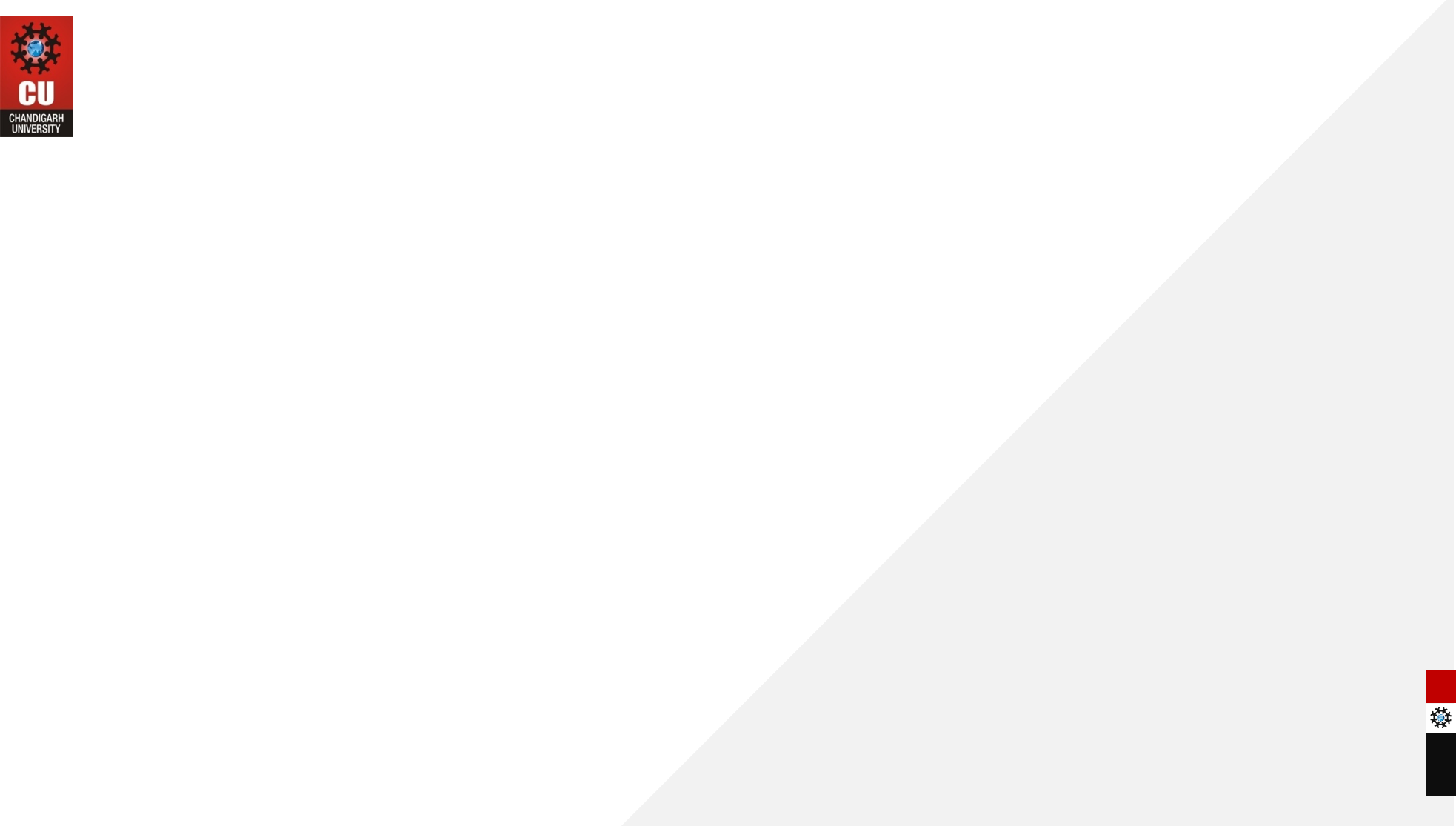
SUBMITTED TO OUR RESPECTED DSA TEACHER

“”MS JYOTI RANI MAM””



# Abstract

* This project simulates a food delivery system using Graphs, Queues, and Linked Lists. It demonstrates how these core data structures can be applied to real-life applications. Graphs model locations and routes, queues handle orders, and linked lists store menus and restaurants dynamically.



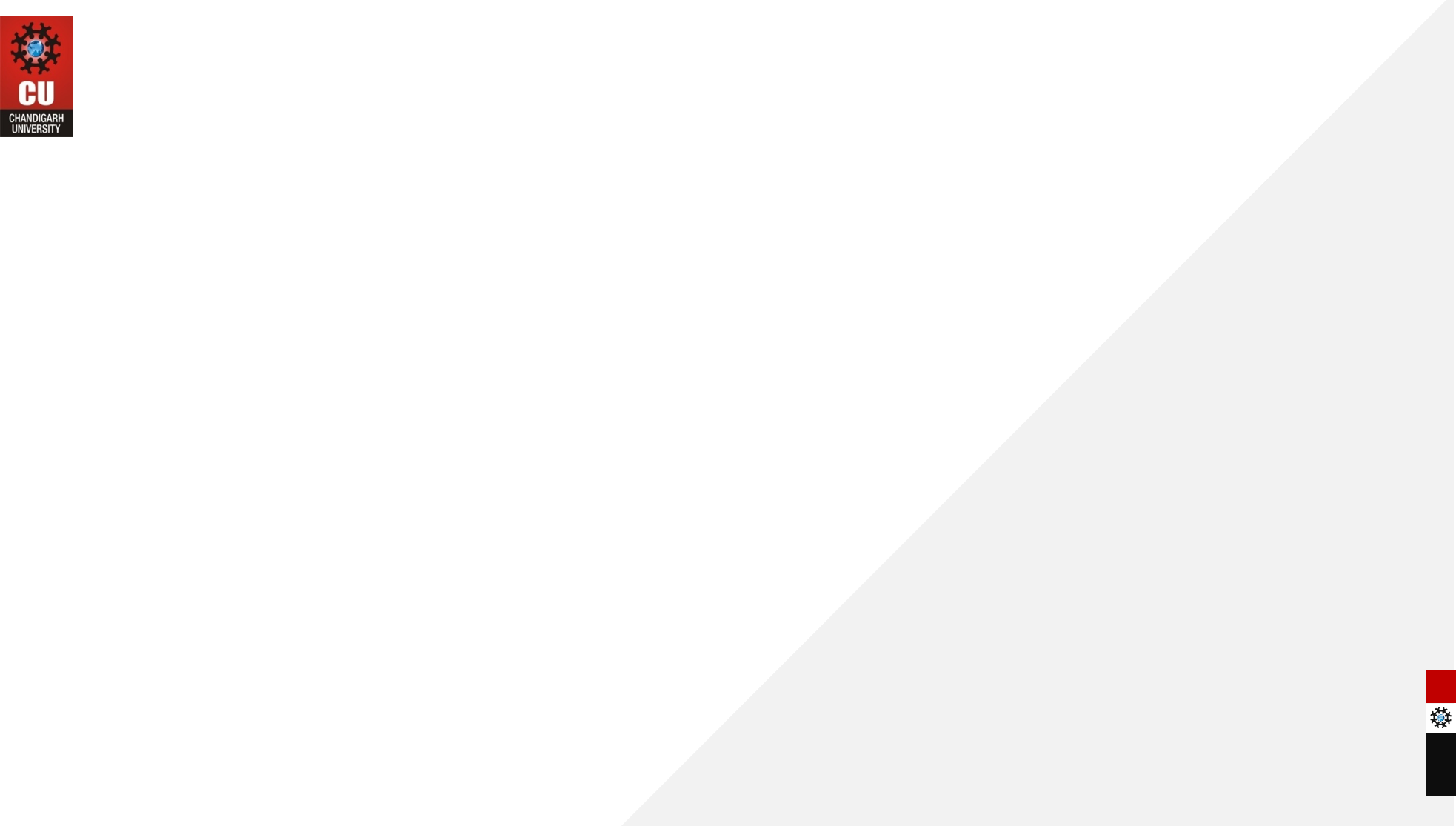
# Project Report

* The system is designed to manage food orders and delivery

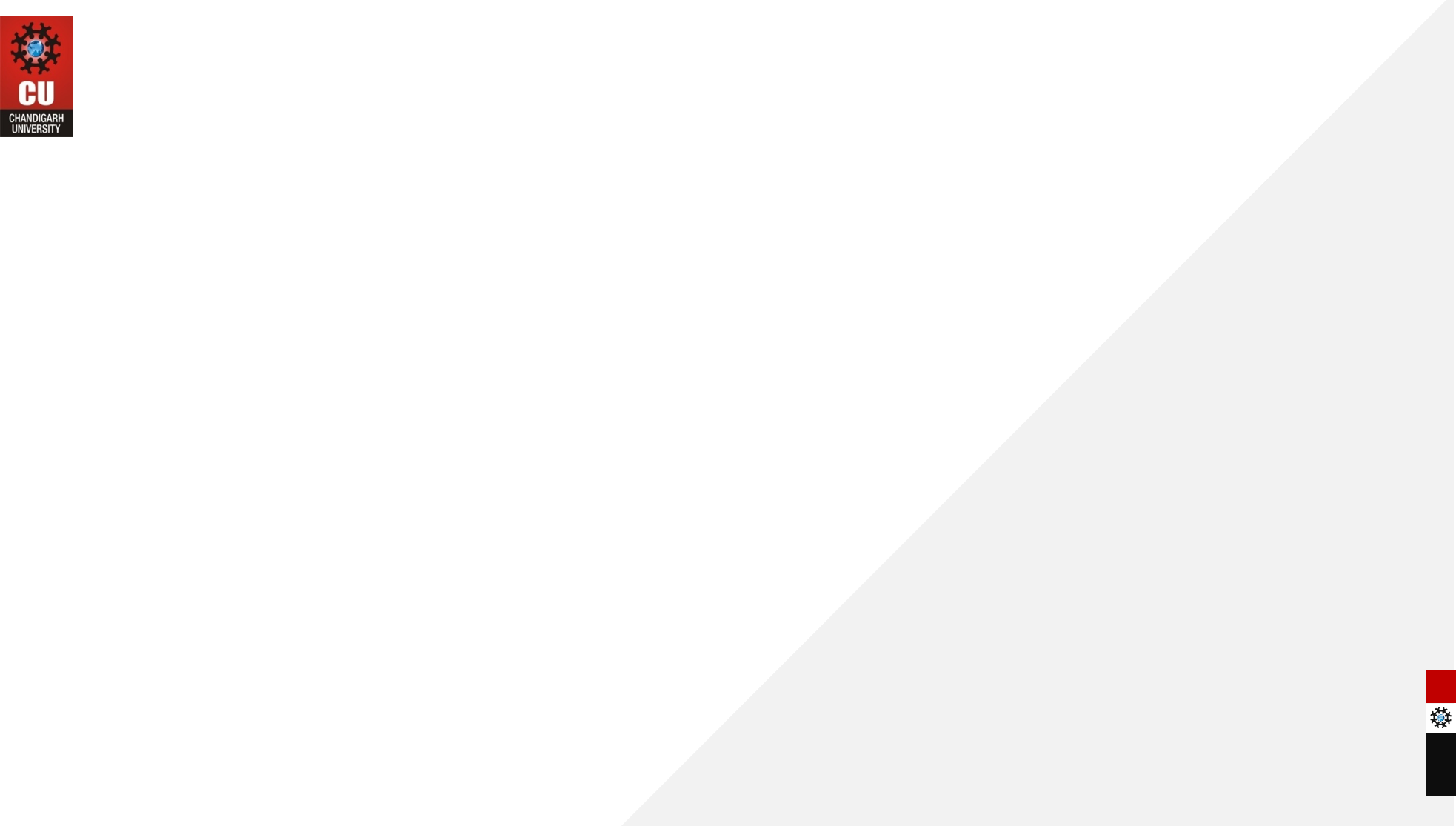
operations efficiently:

* - Graphs represent the city map with locations and shortest delivery paths.
* - Queues manage incoming food orders in a FIFO manner.
* - Linked lists dynamically store restaurant information and menus.
* This combination offers a strong foundation for understanding data

structure integration in real applications.



# Project Code (Sample)

* struct MenuItem {
* char name[30];
* int price;
* struct MenuItem\* next;
* };
* void enqueue(Order order);
* void addRoute(int u, int v, int distance);
* void displayMenu(struct MenuItem\* head);
  1. Menu Management (Linked List) CODE-:

#include <stdio.h> #include <stdlib.h> #include <string.h>

struct MenuItem { char name[50];

int price;

struct MenuItem\* next;

};

struct MenuItem\* addItem(struct MenuItem\* head, char name[], int price) { struct MenuItem\* newItem = (struct MenuItem\*)malloc(sizeof(struct MenuItem));

strcpy(newItem->name, name); newItem->price = price; newItem->next = head;

return newItem;

}

void displayMenu(struct MenuItem\* head) { struct MenuItem\* temp = head;

printf("Menu:\n"); while (temp != NULL) {

printf(" - %s : ₹%d\n", temp->name, temp->price);

temp = temp->next;

}

# Order Queue (Queue) CODE-:

* + - #define SIZE 100
    - struct Order {
    - char item[50];
    - int qty;
    - };
    - struct Order queue[SIZE];
    - int front = -1, rear = -1;
    - void enqueue(char item[], int qty) {
    - if (rear == SIZE - 1) {
    - printf("Queue is full!\n");
    - return;
    - }
    - if (front == -1) front = 0;
    - rear++;
    - strcpy(queue[rear].item, item);
    - queue[rear].qty = qty;
    - }
    - void dequeue() {
    - if (front == -1 || front > rear) {
    - printf("Queue is empty!\n");
    - return;
    - }
    - printf("Processing order: %s x%d\n", queue[front].item, queue[front].qty);
    - front++;
    - }

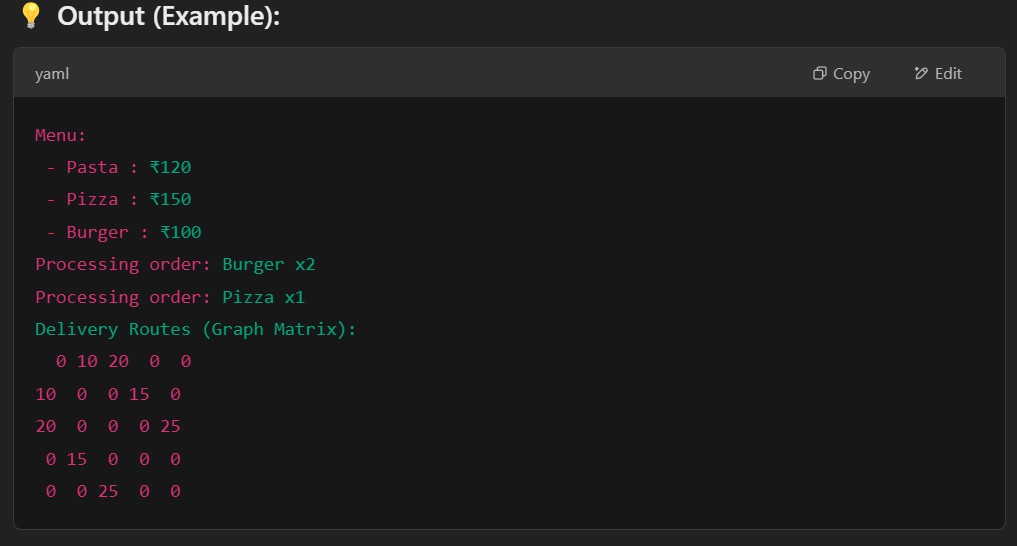
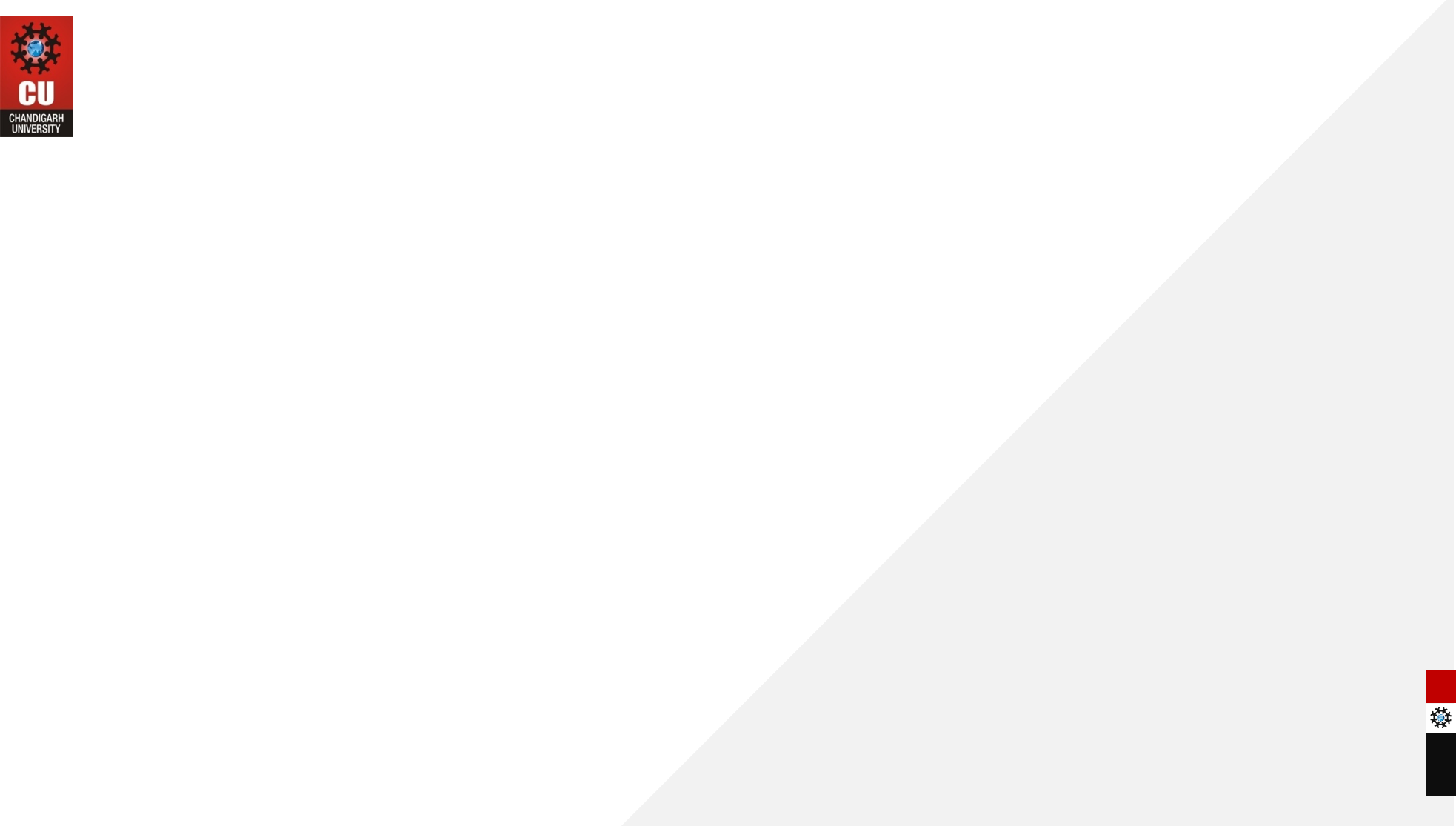
# Delivery Routes (Graph with Adjacency

Matrix) CODE-:

* #define MAX 10
* int graph[MAX][MAX]; // Distance between locations
* int n = 5; // Number of locations
* void addRoute(int u, int v, int dist) {
* graph[u][v] = dist;
* graph[v][u] = dist; // For undirected graph
* }
* void showRoutes() {
* printf("Delivery Routes (Graph Matrix):\n");
* for (int i = 0; i < n; i++) {
* for (int j = 0; j < n; j++) {
* printf("%3d ", graph[i][j]);
* }
* printf("\n");
* }
* }

# Main Function (Integration)

* int main() {
* // Menu setup
* struct MenuItem\* menu = NULL;
* menu = addItem(menu, "Burger", 100);
* menu = addItem(menu, "Pizza", 150);
* menu = addItem(menu, "Pasta", 120);
* displayMenu(menu);
* // Add Orders
* enqueue("Burger", 2);
* enqueue("Pizza", 1);
* dequeue();
* dequeue();
* // Setup Graph
* addRoute(0, 1, 10);
* addRoute(0, 2, 20);
* addRoute(1, 3, 15);
* addRoute(2, 4, 25);
* showRoutes();
* return 0;
* }

Output (Example):