# Discrete Structures

# AI & DS (Fall 2025)

## Instructions:

**You may create more functions and use the provided ones too.**
**Make a group of three and fill in the form on following link. Write group members names in your respective Batch column.**
**https://docs.google.com/spreadsheets/d/1JjPCULeNYk7zVVKghD3SD7wmoMMW9S1MT_aMfnuPJQU/edit?usp=sharing**
**I created 3 separate columns for 3 Batches (AI-1, AI-3 DS-3). Write names in your respective column only. Do not replace/change anyone else' name.**
**Mark your slot by Friday 28th November 2025 evening, after that sheet will be closed and no demo will be conducted for those who didn't marked a slot.**
**Do not change code to read the file. It works correctly.**
**Do not change map.txt file.**
**Project carries 10 absolute marks.**
**Your final marks for project will depend on your demo. If your marks are 0 in demo then entire project marks are 0.**
**Demo will be conducted by your teachers and TAs.**
**Code/Document from AI will result in zero marks and possibly an F grade in course. (Will be checked by Turnitin)**
**Copy/Document code from fellows will result in zero marks and possibly an F grade in course. (Will be checked by Turnitin)**
**Must read submission instructions before submission.**
**Do Not Change Grading Code.**
**Do Not Hard Code Your Outputs as demo will be live.**
**Bring your working code on your laptop in Demo.**
**You can use arrays but do not use any library or any other data-structure.**
**Bonus Marks: (Optional)**
**Display the initial map in better format like an actual graph.**

# Project: Graph Sequence Classification using Cities

## Project Overview

In this project, you will work with a graph representing **cities as nodes** and **connections between them as edges**. Your goal is to implement functions that classify a **sequence of cities** entered by the user as one/many of the following:

- **Walk**
- **Trail**
- **Path**
- **Closed Walk**
- **Circuit**
- **Simple Circuit**

The project will also **calculate the length of the sequence (number of edges)** and allow you to test your functions interactively. You will be graded automatically using a predefined test sequence (During demo we will change it to manually test it).

---

**Files Provided**

1. **main.cpp** – The template code (provided).
2. **map.txt** – Contains city names and their adjacency matrix (graph connections).

---

**Graph Representation**

- Each city is a **node** in the graph.
- The graph is stored as an **adjacency matrix** graph[N][N], where graph[i][j] = 1 if city i is connected to city j.
- City names are stored in a string array cities[N].

---

**Utility Functions**

The template already includes some utility functions. Here's what they do:

1. **getCityIndex(string name)**
    - Input: City name.
    - Output: Index of the city in the cities array.
    - Returns -1 if the city does not exist.
2. **toUpper(string s)**
    - Converts a city name to uppercase for **case-insensitive comparison**.
3. **edgeExists(string a, string b)**
    - Checks if there is a direct connection (edge) between two cities.
    - Returns true if the edge exists, false otherwise.
4. **parseSequence(string input, string seq[])** *(to be implemented by you)*
    - Input: A string sequence entered by the user (e.g., "ISLAMABAD->KARACHI").
    - Output: Populates the array seq[] with city names and returns the number of cities in the sequence.
    - **Example:**
      Input: "ISLAMABAD->KARACHI->LAHORE"
      Output: seq[0] = "ISLAMABAD", seq[1] = "KARACHI", seq[2] = "LAHORE", returns 3.

---

**Student Functions (to Implement)**

You need to implement the following six functions. Each function **returns true or false** depending on whether the sequence satisfies the corresponding condition.

1. **isWalk(string seq[], int len)**
    - Checks if the sequence is a **walk**.
    - A walk is a sequence of vertices where **each consecutive pair is connected by an edge**.
    - A walk can **visit nodes and edges multiple times**.

    - You need to make sure that entered sequence exists in our map as a walk. You will only check remaining if it qualifies for a walk.

2. **isTrail(string seq[], int len)**
    - Checks if the sequence is a **trail**.
    - Make sure it is in our map.
3. **isPath(string seq[], int len)**
    - Checks if the sequence is a **path**.
    - Make sure it is in our map.
4. **isClosedWalk(string seq[], int len)**
    - Checks if the sequence is a **closed walk**.
    - Make sure it is in our map.
5. **isCircuit(string seq[], int len)**
    - Checks if the sequence is a **circuit**.
    - Make sure it is in our map.
6. **isSimpleCircuit(string seq[], int len)**
    - Checks if the sequence is a **simple circuit**.
    - Make sure it is in our map.

---

**Grading System**

- The program automatically grades your implementation using a **predefined test sequence**.
- Each module is graded **0/5** initially.
- If your function returns the **correct result** for the test sequence, it receives **5/5**.
- Total maximum score: **30/30**.

```
--- Module Marks ---
1. Path: 0/5
2. Walk: 0/5
3. Trail: 0/5
4. Closed Walk: 0/5
5. Circuit: 0/5
6. Simple Circuit: 0/5
Total Score: 0/30
```

Once your functions are correct, the output updates automatically:

```
--- Module Marks ---
1. Path: 5/5
2. Walk: 5/5
3. Trail: 5/5
4. Closed Walk: 5/5
5. Circuit: 5/5
6. Simple Circuit: 5/5
Total Score: 30/30
```

**How the Program Works**

1. Reads the **city map** from map.txt.
2. Displays the map with **all connections**.
3. Runs a **predefined test sequence** and grades your functions.
4. Prompts you to **enter your own sequences** for testing.
5. For each sequence, prints:
    - **Length (number of edges)**
    - **Classification**: walk, trail, path, closed walk, circuit, simple circuit

**Sample Output**

```
Enter your own sequence (format A->B->C->...):
ISLAMABAD->KARACHI->LAHORE->PESHAWAR->ISLAMABAD

Length (edges): 4
Sequence classification (Yes/No):
Path: Yes
Walk: Yes
Trail: Yes
Closed Walk: Yes
Circuit: No
Simple Circuit: No
```

---

## Submission Requirements

You need to submit final cpp file and a document.

- Submit your final main.cpp file. Do not change the grading code.
- Your document should contain pre and post conditions for 6 functions that you are implementing. And specify if those functions are in total/partial correctness.
- Your document should also have some output samples of test cases you tried (at least 5 different).
- Make sure to rename your main.cpp file to **your_roll_no.cpp** for example **22i-1990.cpp** and document to **your_roll_no.pdf** for example **22i-1990.pdf** Otherwise, half marks will be deducted.
- Only one submission is required from a group. Do not submit more than one copies.
- Demos will be conducted on Friday, 5th December 2025.
- Deadline is 4th December 2025.
- Submit before due time.
- Submission will close after due date.

You may ask queries to your Teachers Tas or send it to sohail.abbas@isb.nu.edu.pk

*Good Luck*