CMPE 224-343
Fall 2025
Programming Homework 1

This assignment is due by 23:55 on Sunday 26, October 2025.

You are welcome to ask your HW related questions. You should use only one of these options:

1. Moodle Homework **Question Forum**: HW Question-and-Answer (Q&A) Forum on Moodle is always available. Use the "Forum" link at the course Moodle page.
2. Homework **RECITATION HOURS**: There will be two Q&A RECITATION HOURs on the following days:
   - CMPE224-343 HW1 OfficeHour1: October 17 Friday, 06:00-07:00 PM, Zoom ID: https://tedu.zoom.us/j/91488130016?pwd=ILCQhCPYzbRluborNRRTfSKIBqjb2N.1
   - CMPE224-343 HW1 OfficeHour2: October 24 Friday, 06:00-07:00 PM, Zoom ID: https://tedu.zoom.us/j/93845145076?pwd=EekQHm81RI7xXqfWAn1x5eRWfWAPzS.1

Additionally, the tutors will be able to help you understand the homework requirements. Please see the LMS page for their office hours.

You are allowed to complete this assignment **as a group of only 2 people**. If you choose to work with a partner, please **add both of your names** to the Excel file I shared on LMS **until 13 October Monday 23:59**. After this deadline, **no more names can be added**, and you will have to **complete the assignment individually**.

Note: Please make sure that you have read the HW document well before participating. However, no HW related questions will be accepted except from the above options.

PROGRAMMING TASK

In this part, you must implement your own graph data structure by taking inspiration from your textbook and use it to help to solve problem. You are not allowed to use any external library or .jar file. Any solutions without using graph data structure are not evaluated!

Question 1(25 points):

You are working as a systems engineer for a coastal archipelago nation that relies on a complex network of sea bridges to connect its many islands. Each island is considered a **city node**, and each bridge represents a **two-way route** that allows the transport of people, goods, and energy

between islands. Recently, the government noticed that when certain bridges are closed for maintenance or due to storms, **some islands become completely isolated**. They have asked you to design a tool that identifies these **critical bridges** — bridges whose failure would disconnect parts of the country.

Your task is to:

1. Implement a program that identifies all bridges in the island network.
2. For each bridge, print the pair of islands (u, v) in **lexicographic order**.
3. At the end, print the **total number of critical bridges** found.

You must design your own **Graph** class using **adjacency lists**, without using any external libraries or built-in graph packages.

Input:

- The first line contains two space-separated integers: **N** and **M**.
  - **N**: The total number of islands (vertices) in the archipelago.
  - **M**: The total number of two-way bridges (edges) connecting the islands.

- The next **M** lines each contain two space-separated integers **U** and **V**, representing that there is an **undirected bridge** between island **U** and island **V**.
  - The islands are labeled with integers from **0** to **N – 1**.
  - Each bridge allows travel in both directions (U ↔ V).
  - There are **no parallel bridges** (multiple edges between the same pair of islands) and **no self-loops** (a bridge connecting an island to itself).

Sample Input:

```
7 7
0 1
1 2
2 0
1 3
3 4
4 5
5 6
```

Output:

- On the first line, print the text **"Bridge list:"** to indicate the start of the output section.

- On the following lines, print each pair of islands **(u, v)** that represents a critical **bridge** in the archipelago.
  - Each pair must be printed as **two space-separated integers**: `u v`.
  - The pairs should be listed in **lexicographic order** (sorted first by `u`, then by `v`).
  - Each bridge should appear only once, since the graph is undirected.

- On the final line, print the total number of critical bridges found in the network in the following format:
    - Total bridges: K
- where **K** represents the total count of critical bridges detected in the graph.
- If the network contains **no critical bridges**, print only:
    - Total bridges: 0

Sample Output:

```
1 3
3 4
4 5
5 6
Total bridges: 4
```

Question 2(25 points):

You've been hired by the Museum District Council of a historic city to design crowd-flow rules for nightly events. Each museum is a **landmark**, and a pedestrian passage between two museums is **a two-way corridor**. For safety and scheduling, the council wants every nightly route network to be split into two alternating groups—think of them as **Blue and Gold**—so that every corridor always connects a Blue museum to a Gold museum. This prevents bottlenecks and keeps flow balanced.

But the district is large and not all parts of the network follow this rule. Your task is to analyze the entire district and, for each connected sub-network, decide whether this "Blue–Gold alternation" is possible. If a sub-network violates the rule, you must exhibit one concrete contradiction: an odd-length cycle that proves it can't be properly split.

Your task is to:

1. **Detect all connected subnetworks** in the graph.
2. For **each subnetwork**, check whether it is 2-colorable with Blue/Gold using **Breadth-First Search (BFS)**
3. If a component is **not 2-colorable (bipartite)**, output **one odd-length cycle** discovered during BFS as a witness.

Input:

- The first line contains two space-separated integers: **N** and **M**.
    - **N**: Total number of museums (vertices), labeled from **0** to **N – 1**.
    - **M**: Number of two-way corridors (edges).

- The next **M** lines each contain two space-separated integers **U V**, indicating there is an **undirected corridor** between museums **U** and **V**.
    - The graph may be disconnected (multiple components).
    - No self-loops and no parallel edges.

Sample Input:

```
6 5
0 1
1 2
2 0
3 4
4 5
```

Output:

For each connected component (processed in increasing order of its **smallest vertex ID**):

- If the component **is bipartite**, print:
    - Component i: Bipartite
- If the component **is not bipartite**, print:
    - Component i: Not Bipartite
    - Odd cycle: v0 v1 v2 ... v0
- where the cycle is listed **once around** and **starts/ends at the same vertex** to emphasize the cycle.
- Any **valid odd-length cycle** found by your BFS is acceptable.

Sample Output:

```
Component 1: Not Bipartite
Odd cycle: 0 1 2 0
Component 2: Bipartite
```

- **You need to upload your code into VPL on LMS for each question.** If you do not upload your code into VPL on LMS, your homework will **not be graded.**
- The Java sources should be WELL DOCUMENTED as comments, as part of your grade will be based on the level of your comments.
- You need to upload **maximum-3 pages** PDF report document that explains your own answers for programming task in a clearly readable PA report format (refer to **PA REPORT FORMAT** section).

PA REPORT FORMAT

A programming assignment report is a self-description of a programming assignment and your solution. The report must not be hand-written. You may use a word processor or the on-line editor of your choice and prepare as a PDF document. The report must be grammatically correct and use complete English sentences. Each report should include the following sections, in the order given:

Information (%2.5): This section includes your ID, name, section, assignment number information properly.

Problem Statement and Code Design (%15): Include a brief summary of the problem and/or your sub-tasks to be completed in this assignment. You should show your modular design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart using available graphical tools like MS PowerPoint, SmartDraw etc.

Implementation and Functionality (%20): Since you have modular source code, you should describe each sub-module (program) in this section. Each sub-module should include names and types of any input/output parameters as well as the pseudocode algorithm that used for completing its task. By this way, you give meaning to each chart boxes from the previous section.

Testing (%7.5): You should provide a tester class that is able to identify key test points of your program. This class should be able to generate additional (apart from the given sample input/output) test data for the purpose of being clear on what aspects of the solution are being tested with each set. This section should also include a description of any program *bugs* that is, tests which has incorrect results. You should write these to describe your tests, summarize your results, and argue that they cover all types of program behavior.

Final Assessments (%5): In this final section, you should briefly answer the following questions:

1. What were the trouble points in completing this assignment?
2. Which parts were the most challenging for you?
3. What did you like about the assignment? What did you learn from it?
4. Did you use any Artificial Intelligence (AI) tools (e.g., ChatGPT, Copilot, Gemini, etc.) while completing this assignment?
    a. If yes, specify which tools you used and how you used them (e.g., to clarify a concept, check logic, debug, or generate comments).
    b. How did you ensure that your final submission reflects your own understanding and original work?
5. How did AI assistance (if used) affect your learning process?

a. Did it make problem-solving easier, or did it change how you approached the task?
b. What are the ethical boundaries you think should apply when using AI in academic work?

GRADING:

- Codes (%50: %25 for Q1 and %25 for Q2)
  - Available test cases evaluation on VPL: %15
  - Hidden test cases evaluation: %15
  - Approach to the problem: %20
- Report (%50: %25 for Q1 and %25 for Q2)
  - Information: %2.5
  - Problem Statement and Code design: %15
  - Implementation, Functionality: %20
  - Testing: %7.5
  - Final Assessments: %5

IMPORTANT

IMPORTANT NOTES: Do not start your homework before reading these notes!!!

1. This assignment is due by 23:59 on Sunday, October 26th.

2. You should upload your homework to LMS before the deadline. No hardcopy submission is needed. You should upload your codes into VPL and your report into submission place on LMS.

3. The standard rules about late homework submissions apply (20 points will be deducted for each late day). Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.

4. You ARE NOT ALLOWED to modify the given method names. However, if necessary, you may define additional data members and member functions.

5. Your classes' name MUST BE as shown in the homework description.

6. The submissions that do not obey these rules will not be graded.

7. To increase the efficiency of the grading process as well as the readability of your code, you have to follow the following instructions about the format and general layout of your program.

8. Do not forget to write down your id, name, section, assignment number or any other information relevant to your program in the beginning of your Java files. Example:

```
//-------------------------------------------------------
// Title: Scheduler tester class
// Author: Name/Surname
// ID: 2100000000
// Section: 1
// Assignment: 1
// Description: This class tests the …
//-------------------------------------------------------
```

9. Since your codes will be checked without your observation, you should report everything about your implementation. Add detailed comments to your classes, functions, declarations etc. Make sure that you explain each function in the beginning of your function structure. Example:

```
void setVariable(char varName, int varValue)
//-----------------------------------------------------------
// Summary: Assigns a value to the variable whose
// name is given.
// Precondition: varName is a char and varValue is an
// integer
// Postcondition: The value of the variable is set.
//-----------------------------------------------------------
{
      // Body of the function
}
```

10. Indentation, indentation, indentation...

11. This homework will be graded by your TA, Deniz Merve Uzun. Thus, you may ask her your homework related questions through HW forum on Moodle course page. You are also welcome to ask your course tutors Alperen Karadağ, Aysel Arpacı or Mert Temür for help.