Assignment No. 8
EECS 210
Discrete Structures
Due: 11:59 PM, Thursday, December 7, 2023
Submit deliverables in a single zip file to Canvas
Files in other formats (e.g., .tar) will not be graded
Name of the zip file: FirstnameLastname_Assignment8 (with your first and last name)
Name of the Assignment folder within the zip file: FirstnameLastname_Assignment8

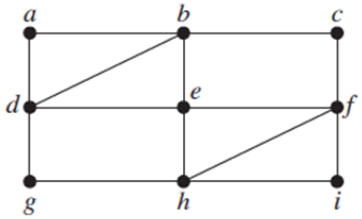
Deliverables:

1. Copy of Rubric8.docx with your name and ID filled out (do not submit a PDF).
2. Source code.
3. Screen print showing the successful execution of your code or copy and paste the output from a console screen to a Word document and PDF it.

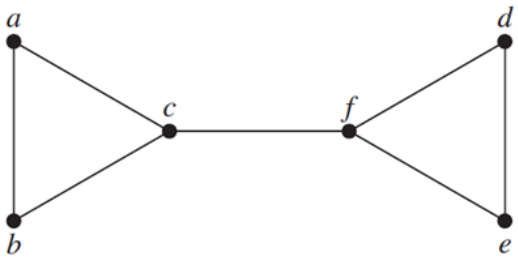
Assignment:

- You may use any language you want, but if you want help from me or one of the SIs, you should probably use C++, Python, or Haskell.
- Your output should make it easy for the grader to determine which part of the assignment it is for.

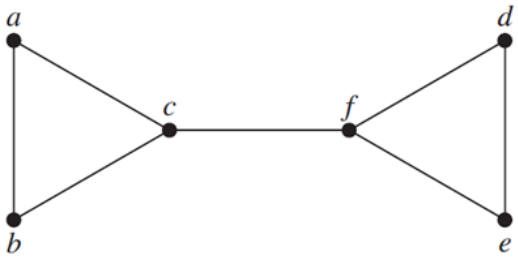
1. Write a program for finding an Euler circuit, if one exists, using the algorithm from the lecture on Euler circuits and paths. If an Euler circuit does not exist, print out the vertices with odd degrees (see Theorem 1). If an Euler circuit does exist, print it out with the vertices of the circuit in order, separated by dashes, e.g., a-b-c.
    a) Debug your program with the Example 1 graphs G_1, G_2, G_3, and the graph of the Bridges of Königsberg from the "Euler, Hamilton, & Shortest Path Problems" lecture slides.
    b) Test your program with this graph:



2. Write a program that uses Dirac's theorem to determine if a graph has a Hamilton circuit. You do not have to find the circuit.
    a) Debug your program with the Example 5 graphs G_1, G_2, and G_3 from the "Euler, Hamilton, & Shortest Path Problems" lecture slides.
    b) Test your program with this graph:



3. Write a program that uses Ore's theorem to determine if a graph has a Hamilton circuit. You do not have to find the circuit.
    a) Debug your program with the Example 5 graphs G_1, G_2, and G_3 from the "Euler, Hamilton, & Shortest Path Problems" lecture slides.
    b) Test your program with this graph:



4. Write a program that creates a min-max strategy for the game of Nim. Display the strategy in a manner that will be clear to the grader and includes all of the elements shown in the "Minmax Strategy for Nim (11.2.5)" slide in the "Application of Trees" lecture.
    a) Debug your program with a version of Nim with the starting position where there are 3 piles of stones containing 2, 2, and 1 stone each, respectively (same as one in the "Minmax Strategy for Nim (11.2.5)" slide.
    b) Test your program with a version of Nim with the starting position consists of three piles with one, two, and three stones, respectively. When drawing the tree, represent by the same vertex symmetric positions that result from the same move.
    c) Write a simulation of two players playing the game of Nim. Player A uses the min-max code you wrote for 4a. Player B makes the next move randomly. Alternate between which player goes first. Play the game 100 times and keep track of who wins and display it. The simulation should show the progression of each game. For example:
        - Start: 2 2 1
        - Player A: 2 1
        - Player B: 1 1
        - Player A: 1 (Player A wins)
5. Provide comments that explain what each line of code is doing. See rubric below.

Rubric for Program Comments		
Exceeds Expectations (90-100%)	**Meets Expectations (80-89%)**	**Unsatisfactory (0-79%)**
Software is adequately commented with prologue comments, comments summarizing major blocks of code, and comments on every line.	Prologue comments are present but missing some items or some major blocks of code are not commented or there are inadequate comments on each line.	Prologue comments are missing all together or there are no comments on major blocks of code or there are very few comments on each line.

Adequate Prologue Comments:

- Name of program contained in the file (e.g., EECS 210 Assignment 3)
- Brief description of the program, e.g.:
    - Python code for demonstrating operations on relations and properties of relations.
- Inputs (e.g., none, for a function, it would be the parameters passed to it)
- Output, e.g.,
    - Print out of the name of each exercise, followed by the exercise's output.
- All collaborators
- Other sources for the code ChatGPT, stackOverflow, etc.
- Author's full name
- Creation date: The date you first create the file, i.e., the date you write this comment

Adequate comments summarizing major blocks of code and comments on every line:

- Provide comments that explain what each line of code is doing.
- You may comment each line of code (e.g., using //) and/or provide a multi-line comment (e.g., using /* and */) that explains what a group of lines does.
- Multi-line comments should be detailed enough that it is clear what each line of code is doing.
- Each block of code must indicate whether you authored the code, you obtained it from one of the sources listed in the prolog, or one of your collaborators authored the code, or if it was a combination of all of these.

Collaboration and other sources for code:

- When you collaborate with other students or use other sources for the code (e.g., ChatGPT, stackOverflow):
    - Your comments must be significantly different from your collaborators.
    - More scrutiny will be applied to grading your comments in particular explaining the code "in your own words", not the source's comments (e.g., ChatGPT's comments).
- Failure to identify collaborators or other sources of code will not only result in a 0 on the assignment but will be considered an act of Academic Misconduct.
- Students who violate conduct policies will be subject to severe penalties, up through and including dismissal from the School of Engineering.