CSCE 221 Cover Page

Programming Assignment#6

Due by April 29st midnight to eCampus

First name: Aldo Last Name: Leon Marquez UIN: 326004699

User Name: allema98 E-mail address: allema98@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more: Aggie Honor System Office

Types of Sources			
People	TA sec 501		
Web Pages	https://piazza.com	Stack Overflow	Geeks for Geeks
Printed material	-		
Other Sources	Lecture Slides		

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.
"On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work."

Name: Aldo Leon Marquez Date: 04/29/2019

Data Structure Description:

The structure implemented for this assignment is a Graph representation using an adjacency matrix. The matrix is created by the provided number of nodes, number of edges, and the edges themselves represented by a pair of nodes. The program gets the graphs from an input. The class created for this is called "euler" and has the following functions and variables:

- vector<vector<int>> Vec; The Adjacency Matrix itself.
- vector<int> edges; Useful vector to test for "bridges".
- int nEdges=0;
- int nNode=0;
- int Odd=0; Variable to store the number of nodes with odd degree.
- int StartNode=0; Starting node of the possible Euler path, default 0.
- euler(string in); Default constructor of a possible Euler path.
- bool isEuler(); Function to check for the possibility of drawing.
- void drawPath(); Function to draw the possible paths.

Conditions to draw with one stroke:

In order to be able to draw with one stroke, i.e visit every edge of the graph exactly once, This conditions can be fulfilled if the graph has an Eulerian cycle or a Eulerian Path.

For an Eulerian Cycle:

- All vertices with non-zero degree must be connected.
- All vertices have even degree.

For an Eulerian Path:

- All vertices with non-zero degree must be connected.
- If and only if there are two odd degree vertices.

Description and runtime of algorithm and functions:

- **euler(string in):** This is the constructor for the graph, inside of it I get the input form a text file. In this function I resize a 2d vector to the appropriate number of nodes $O(nodes^2)$ then the pair of nodes in the text file is used as the index for the matrix and the value is changed from 0 to 1 with runtime $O(edges)$. Another loop is run in the constructor to get a vector that contains the degree of each node, its better to do this from the beginning instead of unnecessarily doing it while drawing the path, the runtime is also $O(nodes^2)$ for this loop so the entire runtime is $O(nodes^2)$.

- **bool isEuler():** This function runs a loop that checks all of the node degrees, it counts the number of odd degrees in the graph and if its something other than 0 or 2, it returns false, otherwise True.
- **void drawPath():** This function is in charge of visiting all nodes and printing the correct Euler path or cycle, The function runs with a $O(edges)$ when ever the graph has a Euler path, if the graph is an Euler Cycle, the function could run an additional n(nodes) times, to get the correct path so the runtime is $O(n * edges)$ for an Eulerian Cycle

Program correctness evidence;

For all of the provided test cases, first the 6 “graph.data” and then the 10 “g.txt”.