MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG315 Algorithms

Assignment 3: Directed Graph of Football Fixtures

This assignment aims to help you practice **graph data structure and basic graph operations**. Your main task in this assignment is to develop a graph of football team victories using **C programming language**. Please note that you need to implement the graph as an **adjacency list** in this assignment.

Overview:

In this assignment, your task is to create a directed graph of football teams based on match data. The information about teams will be provided in a `teams.txt` file, and match data will be in a `matches.txt` file. The goal is to represent teams as vertices and matches as directed edges, with the weight of the edge being the goal difference (ignoring draws).

Internal Processing:

1. Read Data:
    - Read team information from `teams.txt` and match information from `matches.txt`.
    - Create vertices for each team.
    - Create directed edges between winning and losing teams, with the weight being the absolute goal difference.

2. Create Vertex Function:
    - Create a function that takes the graph and team name as input.
    - Create a new vertex for each team and add it to the graph.
    - Each vertex should store the team's name.

3. Create Edge Function:
    - Create a function that takes the graph, winning team, losing team, and goal difference as input.
    - Create a directed edge from the winning team to the losing team with the weight equal to the goal difference.
    - Update the indegree for the losing team and the outdegree for the winning one.

4. Print Graph Function:
    - Print the adjacency list of the resulting graph, showing the vertices and directed edges.
    - Include the number of goals scored for each team.

5. Generate Stats:
    - Print the team with the most wins, the team with the most losses, and the teams with the highest and lowest goal difference respectively.
    - If there is more than one team that have the same stat, you can just print one of them.
    - Note that for each stat, you need to print the team and the statistic itself, for example if Arsenal had the most wins, you would print Arsenal and show how many games they won.
    - Exclude matches that resulted in a draw.

6. Check win chain:
    - Take the name of two teams from the command line.

- Check if there is a win chain between them, meaning if you take teamA and teamB, you need to check if teamA beat a team that beat teamB.

Input:

The program will have two data files as inputs along two command line arguments:
- \`teams.txt\` with team information.
- \`matches.txt\` with match information.

Data Format:
- \`teams.txt\`:

```
Team Name
Manchester United
Arsenal
…
```

- \`matches.txt\`:

```
Season_End_Year;Wk;Date;Home;HomeGoals;AwayGoals;Away;FTR
2016;1;8/8/2015;Manchester Utd;1;0;Tottenham;H
2016;1;8/8/2015;Norwich City;1;3;Crystal Palace;A
2016;1;8/8/2015;Everton;2;2;Watford;D
…
```

Term	**Definition**
Season_End_Year	The year in which the season ended
Wk	The week of the season in which the match took place.
Date	The day the match took place.
Home	The host team.
Away	The guest team.
HomeGoals	Number of goals scored by the hosts.
AwayGoals	Number of goals scored by the guests.
FTR	Result: H if home team wins, A if away team wins, and D if it is a draw.

Functions:
In this assignment, you are given a C file called "template3.c" attached. This will be the template code which you need to strictly follow. Inside the "template3.c" file, all function prototypes are declared, the main function is already written along with a few others. You will basically need to fill in the blanks and write the code for the following functions:

- \`**readTeams**\`: Reads teams data, creating the graph and adding the vertices.
- '**readMatches**': Reads the matches data, creating the edges.
- '**createGraph**': Creates the graph. **(given)**
- \`**createVertex**\`: Creates a vertex for a team. **(given)**
- \`**createEdge**\`: Creates a directed edge between winning and losing teams with the match's goal difference as the weight.
- \`**printGraph**\`: Prints the resulting graph's adjacency list.
- \`**getMostWins**\`: Prints the team with the most wins.
- '**getMostLosses**': Prints the team with the most losses.
- '**getMaxGoals**': Prints the team with the highest total goal difference. The total goal difference for each team is computed by summing the goal differences in the matches the team won and then subtracting the goal differences in the matches the team lost (excluding the matches that resulted in a draw).

- '**getMinGoals**': Prints the team with the lowest total goal difference.
- '**checkWinChain**': checks if the two teams given through the command line have a win chain, meaning whether team 1 beat a team that beat team 2; it will return 1 if yes, 0 if no.
- **'checkPath':** checks if there is a path in the directed graph from team 1 to team 2. it will return 1 if yes, 0 if no.

Important Notes for Implementation

- The template code assumes the data files you are going to read are called "teams.txt" and "matches.txt" and that they are stored directly in the code's immediate directory. Note that if you are using an IDE like Clion, you will need to store the file in the "cmake-build-debug" folder.
- You are not allowed to change any code in the template, and you should abide by the prototypes and structures given. Changing anything in the template may result in an immediate 0 (zero). The places where you need to write your code are marked with the comment: "WRITE YOUR CODE HERE".
- There are data columns in the file that you do not need for your program such as the season end year and weak, so you do not need to consider those data columns.
- You need to ensure usage of comments to clarify your codes.
- Draws are ignored when creating edges.
- You can add helper functions.
- You must make use of createGraph and createVertex in readTeams function, while also making use of createEdge in readMatches function.
- If a team has a space in its name and you want to test it out in checkWinChain function, you simply need to enter its name with an '_' instead of the space in the command line argument and the conversions are already handled in main. For example, for the case of 'Manchester Utd', you simply put 'Manchester_Utd' as the command line argument.

Output:

A sample output with teamD and teamA as command line arguments will look something like the following. **Note that a file containing the actual output when Manchester_Utd and Newcastle_Utd are the command line arguments is attached.**

```
teams.txt has been read successfully; the graph with no edges can be
seen below:
TeamA ->
------------------------------------------------------
TeamB ->
------------------------------------------------------
TeamC ->
------------------------------------------------------
TeamD ->
------------------------------------------------------
*************************************************************

matches.txt has been read successfully; the graph with edges can be
seen below:
TeamA -> TeamB | 1 -> TeamC | 4 -> TeamD | 1
------------------------------------------------------
TeamB -> TeamC | 6
------------------------------------------------------
TeamC ->
------------------------------------------------------
TeamD -> TeamB | 10
------------------------------------------------------
```

```
*************************************************************
The following stats have been generated:
Team with the most victories:
TeamA have won 3 matches

Team with the most losses:
TeamB have lost 2 matches

Team with the highest goal difference:
TeamD have a goal difference of 9

Team with the lowest goal difference:
TeamC have a goal difference of -10
*************************************************************
TeamD have NOT beaten a team that beat teamA.
There is no path from TeamD to TeamA.
```

Incremental and Modular Development

You are expected to follow an incremental development approach. Each time you complete a function or module, you are expected to test it to make sure that it works properly. You are also expected to follow modular programming which means that you are expected to divide your program into a set of meaningful functions.

Professionalism and Ethics

You are expected to complete the assignments on your own. Sharing your work with others, uploading the assignment to online websites to seek solutions, and/or presenting someone else's work as your own work will be considered as cheating.

Rules

- You need to write your program by using C programming.
- You need to name your file with your student id, e.g., 1234567.c, and submit it to ODTUCLASS.
- Code quality, modularity, efficiency, maintainability, and appropriate comments will be part of the grading.
- **You need strictly follow the specifications given in this document.**
- **It is suggested to use helper functions to support modular development.**

Grading Scheme

The assignment will be graded based on the following scheme:

Grading Item	Score
readTeams	5
readMatches	10
createEdge	5
printGraph	10
getMostWins	5
getMostLosses	5
getMaxGoals	10
getMinGoals	10

checkwinChain	20
checkPath	20

Please note that if your function does not produce an expected output, **you will not receive any mark from that function**. You need to ensure your code runs and is testable. **If you submit a code which does not compile, you will automatically get zero.** Please note that code quality, modularity, efficiency, maintainability, and appropriate comments will be part of the grading. Therefore, **providing an expected output does not guarantee a full mark.**