MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG315 Algorithms

Assignment 3: A Simple Scanpath Analyser

This assignment aims to help you practice **graph data structure and basic graph operations**. Your main task in this assignment is to develop a simple scanpath analyser using **C programming language**.

Overview:
Eye tracking is commonly used to understand how people interact with visual stimuli. When people interact with visual stimuli, their eyes become relatively stable at certain points which are referred to as fixations, and the series of these fixations represent their scanpaths. Eye-tracking data analysis is usually conducted based on the areas of interest (AOIs) of visual stimuli, specifically which AOIs are commonly used and in which order. Figure 1 shows an example of a scanpath of a particular user on one web page of Wordpress.com where the red rectangles show the AOIs of the web page, and the yellow circles show the user's fixations - the radius of each circle is directly proportional to the duration of its corresponding fixation.

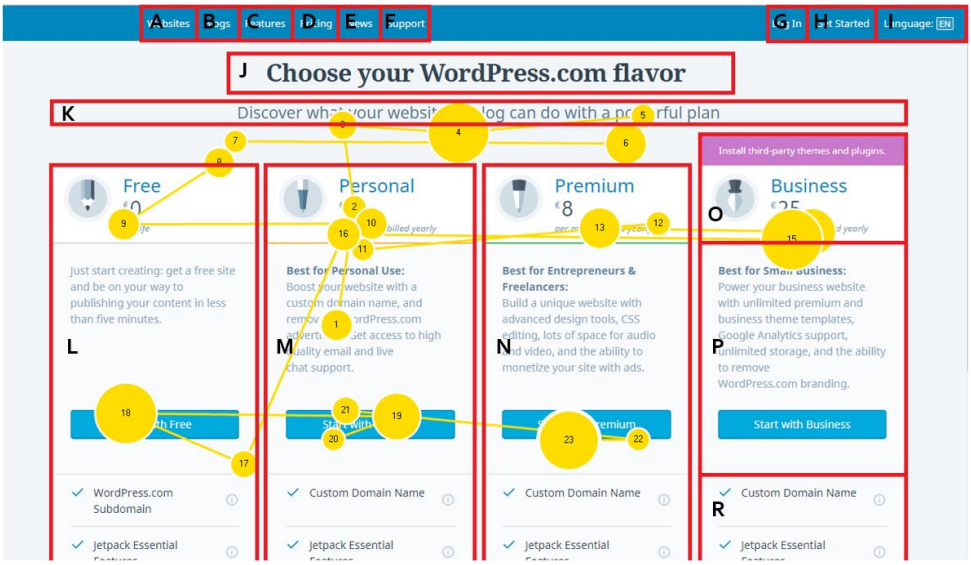

Figure 1. A scanpath of a particular user on one web page of Wordpress.com

Scanpaths are usually analysed based on the AOIs of visual stimuli. They are first represented as string sequences. For example, if a person looks at the AOIs L, M, P, and R respectively, his/her scanpath is represented as LMPR.

In this assignment, you will need to develop a simple scanpath analyser that supports the following functionalities.

1. Read AOIs from a file ("aois.txt") represented in this format: One-character Element Name, Top-left X, Width, Top-left-Y, Height (a separate line for each element). You should not make any assumptions about the number of AOIs. Please note that the coordinate of the top-left point is (0, 0), and it works as shown in Figure 2.

2. Read a series of fixations for several people for a particular visual stimulus from a file ("fixations.txt") where each line represents a fixation in this format: Fixation ID, X-coordinate, Y-coordinate, Duration. Please note that whenever Fixation ID becomes zero, it means that a new person is being read. You should assign a unique id to each person starting from 1. You

should not make any assumptions about the number of people in the file, but you can assume that we can have maximum 100 fixations for a person.

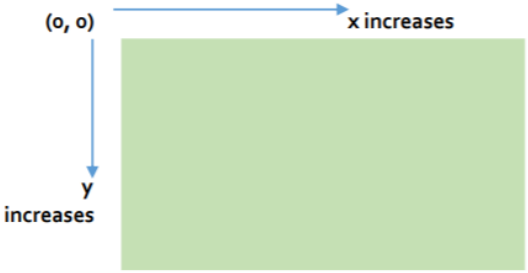

Figure 2. Coordinate System

3. Match the fixations with their respective AOIs and create a scanpath string for each person (such as, ADCB).

4. Create an undirected weighted graph where a vertex is created for each scanpath and an edge is created between two vertices when the corresponding scanpaths have a longest common subsequence (LCS) of at least length 5. The weight of the edge will be equal to the length of the LCS between two scanpaths. An example of a graph is shown below where Scanpath 1 and Scanpath 4 have an LCS of length 5, Scanpath 2 and Scanpath 3 have an LCS of length 6, Scanpath 3 and Scanpath 4 have an LCS of length 5, and Scanpath 3 and Scanpath 5 have an LCS of length 7.

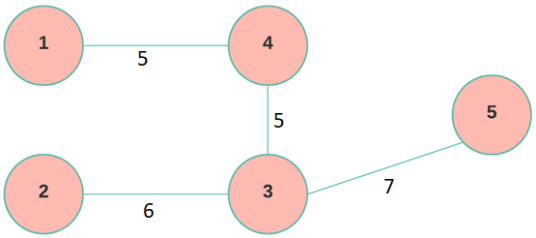

Figure 3. Example Graph

5. Display the graph by showing the vertices and their edges through printing the adjacency list, and display which vertex has the maximum number of nodes adjacent to it and which vertex has the minimum.

Input:
This program will have 2 inputs which is the name of the file containing the information about the AOIs and the name of the file containing the fixations. The input should be taken as follows:

```
Enter AOIs file name > aois.txt
Enter scanpaths file name > fixations.txt
```

Internal Processing:
Once the names of the files are received from the user, the internal processing will be performed in two main steps to create a graph. You will need to implement a separate function for each of these steps, and you can also implement additional helper functions.

- **createVertices:** This function takes the names of the files. It reads the details of the AOIs and the fixations, matches the fixations with their corresponding AOIs and creates a scanpath string for each user. This function returns the graph with a vertex for each scanpath and without any edges. The adjacency list method must be used for graph representation.

- **createEdges:** This function takes the graph created in createVertices. It computes the LCS between each pair of the scanpaths using dynamic programming. If the LCS between two scanpaths has at least 5 characters, the corresponding vertices will be connected with an undirected weighted edge where the weight will be the length of the LCS. The function returns the updated graph.

Output:
Once the graph is created, it should be printed by using the following function.

- **printGraph:** This function will print the adjacency list of the resulting graph along with the vertices with maximum and minimum neighbors respectively. If there are multiple vertices with the same number of maximum or minimum neighbors, it is sufficient to only display one. The format would look like (the order may be different for the edges):

```
The resulting graph's adjacency list:
1 -> 4 | 5
2 -> 3 | 6
3 -> 2 | 6 -> 4 | 5 -> 5 | 7
4 -> 1 | 5 -> 3 | 5
5 -> 3 | 7

Vertex with maximum number of neighbors: 3 has 3 neighbors
Vertex with minimum number of neighbors: 1 has 1 neighbors
```

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. 1234567A3.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.**

Grading Policy
The assignment will be graded as follows:

Grading Item	**Mark (out of 100)**
Graph Representation	10
Main function – coordinate the function calls	10
createVertices	35
createEdges	30
printGraph	15