UNIVERSITY OF PRETORIA
Department of Information Science

IMY 220
Advanced Markup Languages II
Semester Test

Examiner:		11 October 2024
Internal:	Mr ID Bosman	Marks: 24
	Mr M Tarr	Time: 120 minutes
	Mrs T Orsmond	
External:	Mr YL Wong	

This test will run from 07:30 to 09:30. Download the test files from ClickUP. You must submit a zip file with all your files. You also need to add your position, surname and student number to the author field in the package.json. Do not submit this specification.

NB! This section requires you to write functionality in JavaScript. You must, wherever appropriate, make use of the following ES6 syntax:

- Arrow functions
- Appropriate variable declarations
- Template strings
- Object destructuring

Failing to do so will result in a deduction in the marks awarded for each question.

All functions must be written as function expressions. Furthermore, you are **not allowed** to use any loops, *forEach* function, or *$.each* function to implement array-based functionality. You must make use of the other JS array functions as discussed in class. Array-based functionality must work for any number of array items.

Note: wherever you are unable to provide the required functionality in the specified manner, such as not using loops, it is recommended that you focus on providing the functionality. You will receive more marks for working code that does not meet the specifications than for code that does not work at all.

Download *ST1.zip* from ClickUP which contains a package.json which includes all the dependencies you will need to create a React application. Replace the author field in the package.json with your surname, student number and position (SURNAME_POSITION_STUDENT-NUMBER). Your application should include the functionality as detailed below.

The solution for this section is provided in the file *memo.html*. Do not attempt to use the code from this file to create your own solution. If your JS code looks anything like the code in this file, your solution will not be marked.

1.1 Set up a React application using Webpack and Babel as taught in class. You can use the given *package.json* to install all the required dependencies for the test via **npm i**. (4)

1.2 Write a definition for a React class component called **Line**, which gets two points passed in as props. The functionality for working with points should work with any number of dimensions, so you cannot assume, e.g., that there will only be an x and y coordinate for each point. You can assume that the props passed in for both points will always be an array of numbers. (7)

The component must also include a getter for calculating and returning the Euclidean Distance between the points. The formula for calculating this is given below, where a and b are two points and the subscript numbers are the n-th dimensions, e.g., a_1 - b_1 refers to the first dimension in Point a and Point b's coordinates. To make this work for a point with any number of coordinates, you will have to iterate through each Point's coordinates, although you may assume that a Line will always be created with two points with the same number of coordinates.
You must use the JS reduce function to sum together the squares of the differences between the coordinates.

$$d(a,b) = \sqrt{(a_1-b_1)^2 + (a_2-b_2)^2 + \ldots + (a_n-b_n)^2}$$

The Line component should render out each point, the number of dimension of the points (which you may assume is the same per point), and the Euclidean distance of the line as follows:

Point 1: 1;1;1

Point 2: 2;2;2

Dimension: 3

Euclidean Distance: 1.73

1.3 Define valid PropTypes for the Line component and check the following: (2)

- Both point props have been sent through
- Both props are number arrays.

1.4 Write a definition for a React class component called **PointInput** that allows a user to input the coordinates for two points via two inputs. The coordinates should be separated by semicolons (remember that there can be any number of coordinates per point, but you may assume these coordinates will always be numbers, and that each point will have the same number of coordinates). (4)

This component should also include a button that, when clicked, gets the values from each input using React Refs, parses them and then passes these values up to the parent component. Each point contains an array of coordinates in the form of numbers separated by semicolons, e.g., "1;2;3", which needs to be converted into an array of numbers, e.g., [1,2,3].

1.5 Define valid PropTypes for the PointInput component and check the following: (2)

- There is a function being passed down from the parent

1.6 Write a definition for a React class component called **App** that should be the root component of your application. App should use the two child components you have created, Line and PointInput. The App component should handle the state of the application and handle passing data between its children. (5)

At the end, you should have a page that looks similar to the image below