Assignment 3

Exercise 1

a) This line sets the screen orientation for the .MainActivity to "portrait". This means that the activity will always be displayed in portrait mode, regardless of how the device is physically oriented.

The justification for setting the screen orientation to portrait for this app could be due to the nature of the app's user interface or the type of content it displays. For example, if the app is a keyboard app (as suggested by the theme name Theme.AtomikKeyboard), it might be designed to work best in portrait mode where the keyboard layout can take full advantage of the screen's width. This ensures a consistent user experience across different devices and orientations.

b) The keyPresses is not defined as mutableStateOf, because it's not intended to be observed for changes to update the UI. keyPresses is a counter that helps track the number of key presses for logging or other internal processes. Since it doesn't affect the UI, there's no need to declare it with mutableStateOf.

c) In essence, keys.map iterates through the keyboard structure, creating a corresponding set of UI components that represent the keyboard in the Compose view. This transformation allows us to generate the keyboard layout dynamically based on the keys array, facilitating easy updates to the keyboard's structure without manually changing the code.

d) Modifier.weight in Jetpack Compose controls how space is distributed among components in a Row or Column. When used in a keyboard layout, it allows elements to occupy space proportionally based on their weight values.

In the keyboard example:

- Keys have Modifier.weight(1f), meaning they share the available space equally, ensuring consistent key sizes.
- Spacers have different weights (0.5f for half-width, 1f for full-width), allowing for variable spacing between keys.

This approach works for screens of different widths because it creates a responsive and flexible layout, maintaining proportionality regardless of screen size or orientation. It ensures a consistent keyboard appearance across various devices.

e) The height of each Row in Jetpack Compose can be specified using various modifiers. In the context of the keyboard layout, the key modifier that influences row height is Modifier.aspectRatio.

- Modifier.aspectRatio: This modifier sets a fixed ratio between the width and height of a component. In the code, each Key composable uses this modifier with a specific value to maintain a consistent shape (like a square or a hexagon):
    - In the Key composable, the aspect ratio ensures that the key's height is proportional to its width, creating a uniform row height based on the defined ratio.

The aspect ratio determines the height based on the row's width, ensuring that each row maintains consistent proportions, resulting in a balanced keyboard layout.

f) Defining the Key composable as an extension function of RowScope lets it access properties specific to Row, like weight. This is key for components designed to be used within a row context, providing proportional spacing and other row-specific behavior.

If "RowScope." is removed, Key would lose this context, potentially causing errors or unexpected behavior in a Row.

Exercise 2

Results linked in excel file.

c) I think that the results are very close overall.

Exercise 3

a) In Material Design, navigation refers to the system that guides users through the different parts of the app. It helps users understand the app's structure, locate desired features, and move around efficiently.

Destinations and Hierarchy:

- Material Design emphasizes organizing the app's content and tasks into a clear hierarchy. The most important destinations are displayed prominently, while less essential ones are tucked away.

Lateral Navigation:

- Navigation Drawer: Suitable for apps with five or more top-level destinations. It's a panel typically on the left side that slides out to reveal navigation options.
- Bottom Navigation Bar: Provides quick access to 3-5 top-level destinations on mobile devices. Icons represent destinations and stay visible throughout the app, allowing users to switch between them easily.
- Tabs: Used at any level of the app's hierarchy to present related content sets horizontally. They work well for presenting different views of the same data or closely associated functionalities.

Upward Navigation (Going Back Within a Section):

- This allows users to navigate one level upwards within the app's hierarchy. It's crucial to implement consistent upward navigation across all child screens.
- Material Up Action: Used in Android and web apps. It's typically an arrow icon in the top app bar that, when tapped, takes the user back to the previous screen.

b) The top app bar serves multiple purposes related to the current screen:

- Displaying Information: It acts as a prominent location to showcase the app's title, logo, or other relevant information that users should recognize throughout the application.

- Navigation: The top app bar is a prime spot to incorporate navigation elements. We can include buttons or menus that link users to different parts of the app. This can establish a clear navigation hierarchy and streamline user experience.
- Actions: Provide quick access to frequently used actions relevant to the current screen. This could involve icons for search, settings, profile management, or anything that directly affects the content being viewed.
- Branding: By incorporating a logo or design elements consistently within the top app bar, we can create a cohesive user experience.