Task

This is an individual task with a fixed deadline -- no submissions will be accepted after the closing date.

If Turnitin indicates 25% or higher similarity (or a teacher detects a similar level of similarity) with other published work and/or work submitted by other students, the submission will be reviewed and a mark potentially withheld or delayed. Similar sections will **not** be assessed, i.e., if the introduction is copied from elsewhere, the introduction criteria will be awarded 0.

- PDF: A completed report for this task -- the report should set out your development plan/time logs, the key design decisions made, issues encountered and the solutions considered/used, and a reflection on what worked well/what could be improved. Please ensure that explanation/decisions relating to the rubric elements are covered. Any use of genAI must be acknowledged, with prompts and outputs included in an appendix. If genAI was not used, the statement "No Generative AI tools were used for this task" must be included in the acknowledgement. Please refer to the Unit Outline for more details around resubmission/redo of work where the teaching staff feel that genAI has been inappropriately used.
- The report **must** contain a link to code in a repository visible to teaching staff on GitHub Classroom (see GitHub Classroom links for the invite link). Repos are not to be public, nor should tutors be added manually -- **the task will not be assessed until the repo is in a suitable location**.
- Before or after submission, a demo in class is required, otherwise a penalty will be applied. The demo will be assessed according to the Demo guidelines. Note that these will need to be spread over 2 weeks of classes; it is expected that students aiming for high marks will demo in the first week, as this will allow you to take on feedback ahead of submission.

Instructions

1. Start with planning and researching. This will include working through the tutorial exercises and quizzes. Outline and describe the major things you will need to do to create your app.
2. Develop an app that meets criteria as shown below. Do not go overboard -- make use of your class time to clarify requirements if need be.
3. The report needs to cover key knowledge gaps and process that was undertaken in this task, however note that this should be pitched at a fellow student (e.g., "Go to the Android website and download Android Studio" is not required).
4. Submissions must be uploaded as a single PDF.

The app

A local climbing club has a modified climbing wall and scoring process. They are seeking a simple app to score climbs.

- The app needs to have three buttons and one score value at a minimum.

- Set the initial score to 0. When the "Climb" button is clicked, the score increases by a number of points as shown below. When the "Fall" button is clicked, the score decreases by 3. The "Reset" button sets the score back to 0.
- The wall has 9 holds.
    - The climber cannot fall until they have reached the first hold (hold 1). Once the climber has fallen, they cannot climb further.
    - Once the climber has reached the top hold (hold 9), they cannot lose points for falling, and the scoring can only reset from there.
    - When the climber is between holds 1-3, this is the blue zone and each hold scores 1 point.
    - When the climber is between holds 4-6, this is the green zone and each hold scores 2 points.
    - When the climber is between holds 7-9, this is the red zone and each hold scores 3 points.
    - Climbers cannot go backwards.
- The scores should be kept between 0 and 18 (so a fall cannot make the score negative). The score colour should change depending on the zone.
- Two layouts are needed, one for portrait and one for landscape. A different layout type must be used for each.
- The scores must be saved so that on rotation they do not reset. This can be done using saveInstanceState.
- The app must also be usable in a second language.
- Logs must be demonstrated in the code for useful debugging purposes.