University of Westminster
School of Computer Science & Engineering

5COSC023W Coursework 1 (Semester 2)

Module leader	Dr D. Dracopoulos
Unit	Coursework 1
Weighting:	50%
Qualifying mark	30%
Description	
Learning Outcomes Covered in this Assignment:	LO1, LO4, LO5
Handed Out:	February 2026
Due Date	30/3/2026 13:00
Expected deliverables	Source code/XML files/Resources (images, etc)/Video
Method of Submission:	Online via Blackboard
Type of Feedback and Due Date:	Individual feedback via Blackboard within 3 weeks of submission **All marks will remain provisional until formally agreed by an Assessment Board.**

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for Late Submission

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid.

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website :http://www.westminster.ac.uk/study/current-students/resources/academic-regulations

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether

the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Campus Office in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will decide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:**http://www.westminster.ac.uk/study/current-students/resources/academic-regulations**

5COSC023W MOBILE APPLICATION DEVELOPMENT - Assignment 1
Deadline 30/3/2026, 13:00

Dr Dimitris C. Dracopoulos
Email: d.dracopoulos@westminster.ac.uk

Description

You are required to implement an Android application using Kotlin and Jetpack Compose described by the specifications below.

You are not allowed to use third-party libraries. The only libraries that you can use are the standard Android API libraries found in the following URL (with the exception of Views that you should NOT use):

`https://developer.android.com/reference/`

Your are not allowed to use View classes. You should use Jetpack Compose instead.

You should NOT disable `Activity` recreation (e.g. when screen orientation changes) for any parts of your work. You should also not disable re-orientation of the device.

Your task is to implement a cross math puzzle game Android application and according to the game rules and the implementation details below.

The rules of the game

The user is presented with a grid of cells which include mathematical equations placed in the horizontal and vertical directions. Each mathematical equation consists of exactly 2 numbers and one arithmetic operator. Some of the cells are empty and the objective of the game is to fill the blank cells with appropriate numbers so that the equations are satisfied. Calculations are done from left to right and from top to bottom.

For example, if the equation `3 x _ = 33` is present in the puzzle, the user must enter number 11 where the blank `-` is, in order that the equation `3 x 11 = 33` is satisfied.

The arithmetic operations which are allowed to be included in an equation are addition (+), subtraction (-), multiplication (x) and division (/).

Figure 1 shows an example of such a cross math puzzle game.

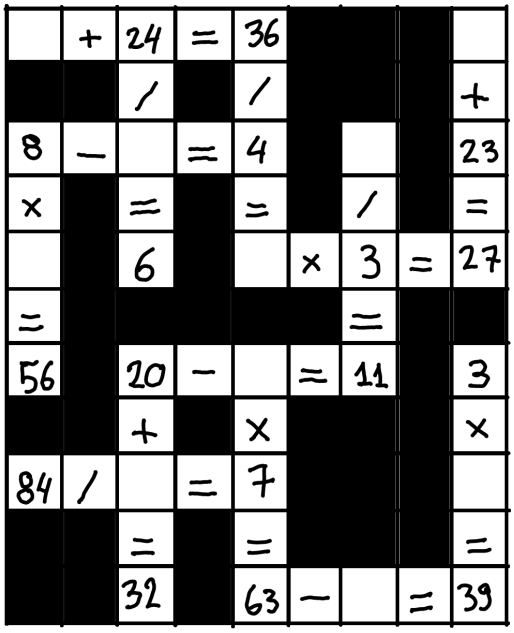

Figure 1: An example screen for the cross math puzzle game. The user just entered 9 for the shown cell and two of the equations were satisfied, therefore their cells become green. The user can still click on the cell occupying 9 and edit it if they wish to do so.

Specification

1. When the application starts, it presents the user with 3 buttons labelled *New Game*, *Advanced Level* and *About*. *(2 marks)*

2. Clicking on the *About* button should present the user with a popup window which describes the author (student id and name) and the message:

   ```
   I confirm that I understand what plagiarism is and have read and
   understood the section on Assessment Offences in the Essential
   Information for Students.  The work that I have submitted is
   entirely my own. Any work from other authors is duly referenced
   and acknowledged.
   ```

   (3 marks)

3. Clicking on the *New Game* button, the user will be presented with the game screen which they interact with (see Figure 1 for an example of such a screen).

   Each time the user clicks on an empty cell, they are presented with a way to input a number to occupy that cell (it is up to you to determine the way to enter input by either a popup window or some other alternative input method).

   Every subsequent press of the *New Game* button should display different arithmetic equations (numbers and arithmetic operators) with the blank cells located in different parts of the equations. E.g. in one instance of clicking the *New Game* button the equation `3 x _ = 33` could be displayed horizontally on the top left corner of the grid, while in another instance of clicking the *New Game* button the equation `10 / 2 = -` might be displayed vertically on the top left corner of the grid.

   The user should be able to click and edit a cell which was previously empty but now it is occupied by a number entered by the user. In such a case, the user can change the number that they previously input for that cell. It is not possible to change the numbers or arithmetic operators that were displayed when the initial grid was presented to the user after clicking the *New Game* button.

   (19 marks)

   - 4 marks for displaying the grid with equations
   - 4 marks for displaying valid equations
   - 4 marks for displaying different equations each time the button is clicked
   - 3 marks for allowing the user to enter a number for an empty cell and which subsequently occupies the cell
   - 4 marks for the ability to change (edit) a number that was previously entered by the user.

4. Every time that the user enters a number for an empty cell and one of an equation displayed is satisfied (i.e. there are no empty terms in the displayed equation), all the cells which are part of the equation are highlighted as green colour (see Figure 2. If the user enters a number which does not satisfy the corresponding equation (and there are no more empty cells which are part of the equation), then all the cells of the equation are highlighted as

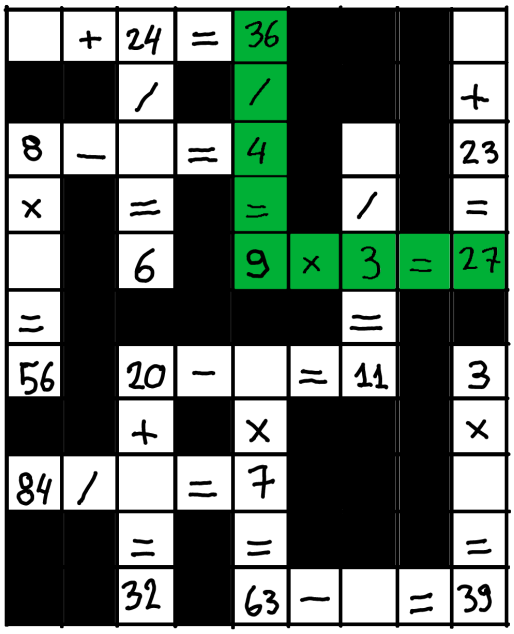

Figure 2: An example screen for the cross math puzzle game. The user needs to fill in all the blanks correctly in a way that all the equations are satisfied.

red. In both cases, the cell that the user clicked to enter a number is still clickable, in case they wish to change the numbers input.

(10 marks)

- 4 marks: Correct Red/green colours
- 3 marks: Updating all equations including the empty cell, if that cell is involved in more than 1 equation
- 3 marks: Changing a number by the user refreshes the colours appropriately, i.e. from green to red or from red to green.

5. Every time that the user completes an equation correctly they get 1 point for each equation satisfied. I.e. if an empty cell is included in 2 equations and the entered number satisfy both of the equations then 2 points are scored.

   The current total score for the game is displayed on the top right of the screen.

   (5 marks)

   - 2 marks for the score being displayed on the top right corner.
   - 3 marks for correctly updating the score

6. Extend the application so that every time that the *New Game* button is clicked, a different number of equations is displayed in a different grid size. The terms of the equations are random as it is the location of an empty cell in an equation. The maximum size of the grid is 20 rows and 20 columns. The minimum number of rows is 11 and the minimum number of columns is 11. The number of rows in the grid does not have to be always the same with the number of columns.

   (12 marks)

   - 5 marks for displaying different grid sizes
   - 4 marks for displaying a different number of equations
   - 3 marks for a random location of the empty cell in an equation.

7. As soon as the full puzzle is filled correctly, the game is not playable any more. In order to start a new game, the user needs to press the Android "Back" button to move to the initial screen of the application from where a new game can be started by pressing the *New Game* button.

   (3 marks)

8. Clicking on the *Advanced Level* button in the initial screen of the application, the user will be presented with a math cross game screen which contains as many empty cells as possible as long as the puzzle is solvable (see for example Figure 3). The filled puzzle in this particular instance (i.e. the solution) is shown in Figure 4.

   If you implement this subquestion, you should make sure that you describe in detail the approach that you come up with, its justification and why do you think that it will perform well, advantages and disadvantages. The documentation of this should be done inside comments in the very beginning, or the very end of your implementation of this part.

   (17 marks)

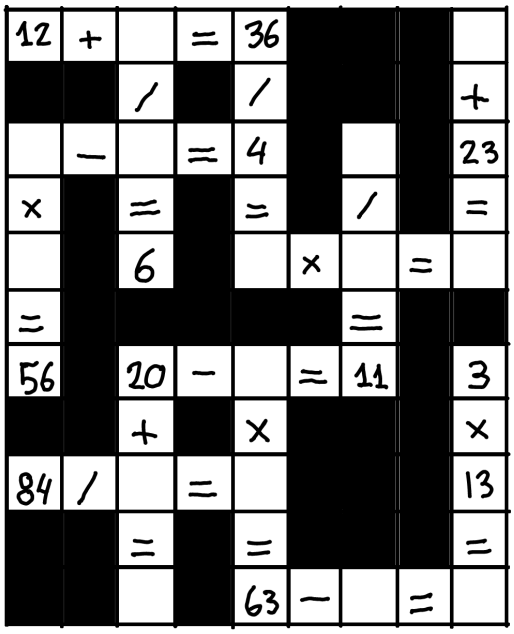

Figure 3: An example screen for advanced level of the cross math puzzle game. Notice that there are more than 1 blank cells as part of some equations but the puzzle can still be solved.

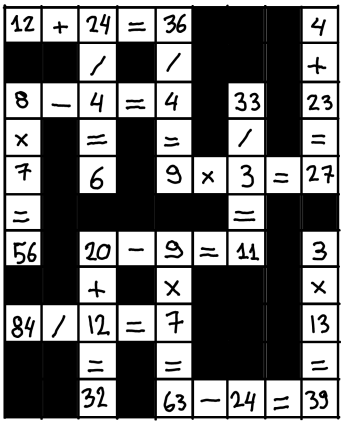

Figure 4: The solution to the cross math puzzle game shown in Figure 3.

- 9 marks for devising an efficient algorithm
  - 8–9 marks for excellent derivation of a strategy
  - 6–7 marks for minor omissions of a good approach
  - 4–5 marks for some kind of strategy which partially works
  - 1–3 marks for a poor choice of a strategy
- 8 marks for proper documentation and justification
  - 4 marks for documentation
  - 4 marks for justification

9. Extend your implementation by allowing the human player to set the number of equations that the puzzle game displays.

   (3 marks).

10. Extend your implementation so that the game screen contains a `Switch` component. The user can toggle on or off the switch. Every time it is switched on, a countdown timer, counting from 60 down to 1, every tick occurring after 1 second exactly. The countdown timer is displayed on the top left of the screen. As soon as the counter reaches the value of 0, the current game screen stops (i.e. the user has 60 secs to complete the puzzle) and the message *GAME OVER!* is displayed to the user. After this point the game is not playable any more (i.e. if the user clicks on any of the empty cells nothing happens).

    (9 marks).

    - 3 marks for displaying the timer with a countdown.
    - 2 marks for implementing the switch component.
    - 3 marks for the game not being playable any more after the timer reaches 0.
    - 1 mark for the *GAME OVER!* message.

11. For all the tasks, the application should behave in a user friendly manner when the device is rotated from portrait to landscape and back to portrait mode. I.e. the application should resume from exactly the same point (same screen and data) when the orientation changes. The rotation of the device should not change what was the user was seeing before the rotation.

    You should NOT disable `Activity` recreation (e.g. when screen orientation changes) for this or any parts of your work. You should also not disable re-orientation of the device.

    (9 marks)

    - 1-5 marks: partial implementation
    - 6-9 marks: minor omissions

Marking Scheme: The marks achieved for each part of the program are indicated in the description of the task above. In addition to these the following will be taken into account:

- *Code readability* (structure, comments, variable naming, etc,): 4%
- *Implementation* (e.g. quality, efficiency, look and feel of the application, based on fonts, colours, etc.): 4%

In addition to the above marks indicated in each of the sub-questions in the specification, additional marks will be deducted if an application behaves in an unexpected way. For example, an application should not crash, the application should work properly even if a user enters invalid data or rotate the device.

The maximum for work which does not compile (or XML or other resources files with syntax errors causes the project not to build) is 30%.

Failure to submit a video or a working link to a video will result to a capping of your mark to 30% (i.e. if your submitted work is given initially a mark above 30% it will be capped to 30%).

Submission of assignments using a different method other than Blackboard will not be accepted and zero (0) marks will be awarded in such cases. The exception is a working link to a video outside Blackboard.

Deadline: Monday 30th of March 2026, 13:00.

Submission Instructions

Files to submit:

1. All of the files of the Android Studio project of your application in a zip file.
2. A video demonstrating all the functionality that you implemented for your application. If the video size exceeds the size allowed by Blackboard you can submit a link to a video to another site (e.g. Google Drive, Youtube, etc.) where you have uploaded the video. Make sure that if your video is located to another web site and not Blackboard that you have given appropriate permissions to external people to access your video.

   Video Guidelines:

(a) Try to keep it as short as possible to demonstrate all of the functionality that you implemented. It should not be more than a few minutes.

(b) You should not explain any code just show the functionality.

(c) Audio should accompany the demo of the functionality, e.g.. "I press this button and this happens"

(d) Make sure that if you submit the video in an external site (youtube, etc) everyone has permissions to watch it. Otherwise, it will count as no submission of the video. If you included the video as part of your Blackboard submission this does not apply.

(e) It would be helpful if you state which tasks you did not implement at all or implemented partially.

Therefore an example of the format could be:

(a) Here is task 1, I press this and this happens. I rotate the device and this happens. etc....

(b) Here is task 2, ...

(c) I did not attempt task3.

(d) I partially implemented task 4, this is what I have.

(e) I did not implement tasks 5, 6, 7, 8.

Referencing code: Any code taken from other resources (i.e. a textbook or internet) should be referenced in comments within your code (full textbook details or full web URL), identifying the exact code that you used it as part of your application and the exact portions of the original source code that you reused.

You should submit via BlackBoard's Assignment functionality (do NOT use email, as email submissions will be ignored.), all the files described above. A single zip file with the name `wNNNNNNNN` (where `wNNNNNNN` is your university ID login name) containing all the above files could be submitted alternatively. You can create such a file by using the main menu in Android Studio and choose `File->Manage IDE Settings->Export to Zip File...`

Note that Blackboard will allow to make a submission multiple times. Make sure before submitting (i.e. before pressing the `Submit` button), that all the files you want to submit are contained there (or in the zip file you submit).

In the case of more than one submissions, only your last submission before the deadline given to you will be marked, so make sure that all the files are included in the last submission attempt and the last attempt is before the coursework deadline.

Request to mark submissions which are earlier than the last submission before the given deadline will be ignored as it is your responsibility to make sure everything is included in your last submission.

The following describes how to submit your work via BlackBoard:

1. Access `https://learning.westminster.ac.uk` and login using your username and password (if either of those is not known to you, contact the Service Desk, tel: +44 (0) 207 915 5488 or log a call via `https://servicedesk.westminster.ac.uk`.).

2. Click on the module's name, `MODULE: 5COSC023W.2025 MOBILE APPLICATION DEVELOPMENT` found under `My Modules & Courses`.

3. Click on the `Assessment->Submit Coursework->Coursework`.
4. Click on `View Assignment`.
5. Attach your zip file containing all the Kotlin source code files and resources of the Android Studio Project, by using the `Browse` button.
6. Attach your video or include a link of your video as the first comment line) in the MainActivity of your code.
7. Create a Word or PDF file with the following information:
   - *Comments*: Type your full name and your registration number, followed by: "I confirm that I understand what plagiarism is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged."
8. Attach the file with the statement above.
9. Check that you have attached both the zip, the video (or the video link) and the statement file.
10. Click the `Submit` button.

If Blackboard is unavailable before the deadline you must email the Registry at `studentcentre@westminster.ac.uk` with **cc:** to myself and your personal tutor before the deadline with a copy of the assignment, following the naming, title and comments conventions as given above and stating the time that you tried to access Blackboard. You are still expected to submit your assignment via Blackboard. Please keep checking Blackboard's availability at regular intervals up to and after the deadline for submission. You must submit your coursework through Blackboard as soon as you can after Blackboard becomes available again even if you have also emailed the coursework to the above recipients.