University of Westminster
Department of Computer Science

ECWM511 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:	2/2/2016
Due Date	7/3/2016 10:00am
Expected deliverables	Source code/XML files
Method of Submission:	Online via Blackboard
Type of Feedback and Due Date:	Individual feedback verbally straight after the viva and written individual feedback via Blackboard within 2 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**

ECWM511 MOBILE APPLICATION DEVELOPMENT - Assignment 1
Deadline 7/3/2016, 10:00am

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

Description

You are required to implement an Android application described by the specifications below. The application will be a basic *brain training game*, which will be asking the user to calculate the answer to various simple arithmetic expressions.

It is important to follow exactly the specifications and your implementation must conform to these:

1. When the application starts, it presents the user 4 buttons labelled *New Game*, *Continue*, *About* and *Exit*. *(7 marks)*

2. Clicking on the *About* button, it should present the user with a popup window which describes the rules of the game (it is left up to you to determine the appropriate text of the rules displayed, by looking at the subquestions below). *(2.5 marks)*

3. Clicking on the *Exit* button should terminate the application. *(2.5 marks)*

4. Clicking on the *Continue* button, a previously started game is resumed. The game resumes from exactly the same point that it was left before, i.e. the implementation needs to restore the exact previous state of the game.

   For this, in order to work, the user should be asked whether he would like to save the current game when he clicks on the *Exit* button. Next time the application starts and the user clicks on the *Continue* button, the state of the last game saved should be restored. *(5 marks)*

5. Clicking on the *New Game* button, starts a new game for the user.

   The user is presented with four levels of game difficulty, *Novice*, *Easy*, *Medium*, *Guru*. Clicking on each one of them sets the maximum number of terms involved in the arithmetic expression that the user needs to calculate (see the next subquestion).

   The *Novice* level corresponds to an arithmetic expression involving exactly 2 integers. The *Easy* level corresponds to an arithmetic expression with maximum 3 terms, i.e. it can contain 2 or 3 integers. The *Medium* level corresponds to an arithmetic expression with maximum 4 terms, i.e. it can contain 2, 3, or 4 integers. The *Guru* level corresponds to an arithmetic expression with maximum 6 terms and minimum 4 terms, i.e. it can contain 4, 5 or 6 integers. *(5 marks)*

6. After the user sets the difficulty level, by clicking on one of the 4 levels, the user is presented with the main **game screen** which contains a random (not fixed or hardcoded but randomly generated by the program itself) arithmetic expression among integers, based on the difficulty level he/she has chosen.

   The arithmetic expression includes random mixed operations utilising addition, subtraction, multiplication and division.

   Following the above description, if the user has chosen previously a *Medium* level of difficulty, he/she will be presented with a single random arithmetic expression which includes 2, 3, or 4 integers. Examples of such expressions are $55 * 2 - 4 - 101$, $102/3/3$, $589 - 281$, $123 + 5 * 6 + 2$.

   Besides the arithmetic expression, the **game screen** also contains 13 buttons corresponding to numbers 0 to 9, the `DEL` key, the # key and the minus − key. The screen should be looking exactly like Figure 1, although spacing, fonts, colours and sizes of boxes and buttons can be different.

   Once the user starts typing the correct answer by using the game screen buttons, the question mark **?** will disappear and replaced with the numbers that the user types. E.g. if the user enters number 1 followed by 3, etc. then the numbers 1, 3, etc. should replace the question mark in the Figure. A digit appears on the screen as soon as the user types each single digit, by using the on screen buttons, i.e. the answer appears digit by digit and NOT just when the user types the whole answer. If the user presses `DEL` and deletes all digits typed, the **?** does not appear again.

   The user indicates that the full answer is completed by hitting the # button. If the user makes a mistake he/she can delete the last digit typed by pressing the `DEL` button.

   The arithmetic expression should be evaluated from left to right and all operations assume the same precedence. No parentheses or other brackets should be presented to the user to determine the precedence. Every arithmetic expression (including subexpressions within the whole expression) should be evaluated in such a way, that if the result is a non-integer, the result should be rounded to the closest integer number. For example, the expression 10/3*2 should evaluate to 6, because 10/3 is 3.333 which is rounded to 3, then multiplied by 2, giving the overall result of 6.

   In total, for a single game, the user will be presented and asked to guess 10 different arithmetic expressions.

   Note that this subquestion, requires the user to use the game screen buttons to give the answer and NOT the Android device soft or hard keyboard!

   (28 marks)

7. As soon as the user enters the # sign following the answer, the message *CORRECT!* (in green colour) or the message *WRONG!* (in red colour) appears on the screen, depending on whether the answer given is correct or incorrect respectively, as shown in Figure 2. Pressing the # button again, presents the user with the next arithmetic expression, until the user completes the full cycle of guessing 10 arithmetic expressions. *(7 marks)*

8. Extend the implementation so that the user can switch on hints from a preference menu. If the hints option is "on" then he/she will be given 4 attempts per question (arithmetic expression), in case his first, second and third attempts to answer are incorrect. Every time that the user gives an incorrect answer when this option is "on" the application will

be displaying "greater" (if the user's answer is less than the correct answer) and "less" (if the user's answer is greater than the correct answer). *(8 marks).*

9. Extend the application by providing a countdown timer, counting from 10 down to 1, every tick occurring after 1 second exactly. The countdown timer is displayed in the main screen of the game, as shown in Figure 3.

   As soon as the counter reaches the value of 0, the next arithmetic expression is presented to the user (in this case the user does not need to press the # to move to the next question). However, if the user presses the # button before the counter reaches 0, the counter stops, the game displays whether the answer is correct or incorrect and a subsequent press of # moves to the next question.

   In the case that hints are "on", a press of the # button before the counter reaches 0 does not stop the counter if the answer is incorrect. The counter keeps running until it reaches 0, or until the maximum attempts per question is reached (4) or until the user gives the correct answer to the question. *(15 marks)*

10. In the end of each game (after 10 arithmetic expressions), the user will be displayed with a score which the sum of the points scored in each question. The score will take into account how fast the user answered the question. It will be calculated based on the following:

    (a) 0 marks for each incorrect guess

    (b) $\frac{100}{10-time\_remaining}$ marks for each correct guess. $time\_remaining$ is the value of the countdown timer when the user pressed the # button to submit the answer. For example, if the user answered a question correctly and the time remaining was 4 secs, then the points received for the question are 100/(10-4) = 100/6=17 points. Points are rounded to the closest integer value

    We assume that the user usually cannot answer in less than 0 secs in order to avoid a denominator of 0. If the user answers a question with 10 secs remaining, then the points awarded are 100 (for a correct guess), i.e. we don't use the above formula.

    (10 marks)

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,): 5%
- *Implementation* (e.g. quality, efficiency, look and feel of the application, based on fonts, colours, etc.): 5%

The maximum for work which does not compile (or XML files with syntax errors causing the Java code not to compile) is 30%.

Based on the functionality implemented, the marks awarded will consist of 2 parts:

- 30% of the marks achieved will be awarded based on the submission.
- The remaining 70% of the marks for the implementation will be awarded after a compulsory viva, that will test the understanding of the code by the student. The student will be asked to demonstrate the application and will be asked questions about the code to demonstrate his/her understanding.

A compulsory viva for each student based on his/her submission will take place during the next tutorial session after the submission (all of them taking place in the weeks starting the 7th of March and 14th of March). Each student will be notified a specific slot that he/she needs to attend. Failure to turn up in the viva slot designated (no changes will be allowed as this a normal examination) will result in awarding only 30% of the marks achieved for the submission (see marking scheme above)

It is the responsibility of each student to make sure that during the viva the code runs properly in the lab used during the viva, i.e. you should make sure in advance (allow enough time before the viva day) that everything is running properly in the machine you will be using. If you developed code at your home computer, it is your responsibility that you port it to the lab in advance, before the viva. Marks will be awarded based on the demo/viva and excuses of the type "it used to run - don't know what happened since last time" will not be accepted or awarded with extra marks.

You are allowed to use your own laptop during the viva if you wish to.

Submission of assignments using a different method other than Blackboard will not be accepted and zero (0) marks will be awarded in such cases.

Deadline: Monday 7th of March 2016, 10:00am.

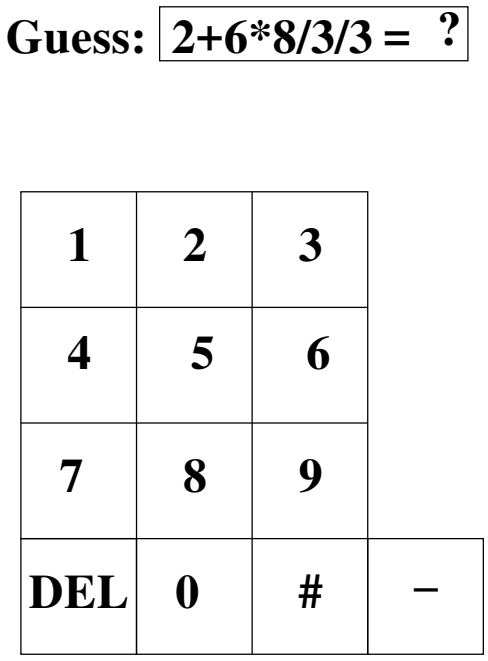

Figure 1: Main screen of the brain training game.

Submission Instructions

Files to submit: All of the files of the Android Studio project of your application in a zip file.

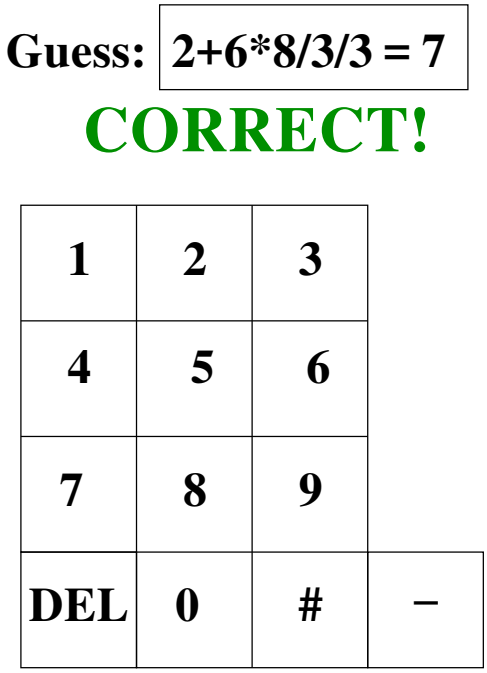

Figure 2: Main screen of the brain training game. User has typed a correct answer.

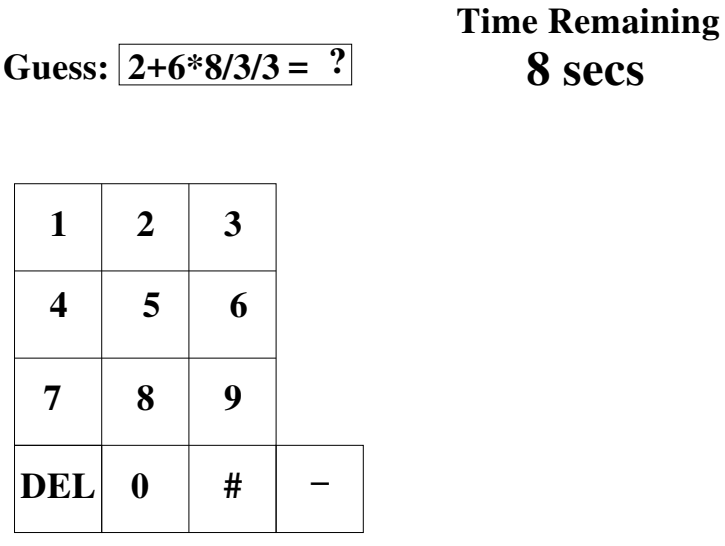

Figure 3: Main screen of the brain training game with a countdown timer.

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 containing all the above files could be submitted alternatively.

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, ask the HelpDesk at the Library.).
2. Click on the module's name, `MODULE: ECWM511.2016 MOBILE APPLICATION DEVELOPMENT` found under `My Modules & Courses`.
3. Click on the `Assignments` button found on the left hand side menu.
4. Click on `View/Complete Assignment`.
5. Attach your zip file containing all your files of your Android project, by using the `Browse` button.
6. Fill in the requested 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."
7. Click the `Submit` button.

If Blackboard is unavailable before the deadline you must email `FSTRegistry@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.

Coursework Marking scheme

The Coursework will be marked based on the following marking criteria:

Criteria	Mark per component	Mark provided	Comments
Implementation	**100**		
Functionality	90		For a split of the marks see the subquestions description in the main description of the coursework
Code Readability	5		structure, comments, variable naming, etc,
Software Quality	5		Quality, efficiency, etc.
Total	**100**		