COMP1551 (2018/19)	**Application Development**	**Faculty Header ID: 300482**	**Contribution: 100% of course**
Course Leader: **Dr Elena Irena Popa**	**Coursework**		**Deadline Date:** **Monday 09/12/2019**
This coursework should take an average student who is up-to-date with tutorial work approximately 50 hours Feedback and grades are normally made available within 15 working days of the coursework deadline			
Learning Outcomes: A,B,C			

Plagiarism is presenting somebody else's work as your own. It includes: copying information directly from the Web or books without referencing the material; submitting joint coursework as an individual effort; copying another student's coursework; stealing coursework from another student and submitting it as your own work. Suspected plagiarism will be investigated and if found to have occurred will be dealt with according to the procedures set down by the University. Please see your student handbook for further details of what is / isn't plagiarism.

All material copied or amended from any source (e.g. internet, books) must be referenced correctly according to the reference style you are using.

Your work will be submitted for plagiarism checking. Any attempt to bypass our plagiarism detection systems will be treated as a severe Assessment Offence.

Coursework Submission Requirements

- *An electronic copy of your work for this coursework must be fully uploaded on the Deadline Date of **Monday 09/12/2019** using the link on the coursework Moodle page for COMP1551.*
- *For this coursework you must submit a single PDF document. In general, any text in the document must not be an image (i.e. must not be scanned) and would normally be generated from other documents (e.g. MS Office using "Save As .. PDF"). An exception to this is hand written mathematical notation, but when scanning do ensure the file size is not excessive.*
- *For this coursework you must also upload a single **ZIP** file containing supporting evidence.*
- *There are limits on the file size (see the relevant course Moodle page).*
- *Make sure that any files you upload are virus-free and not protected by a password or corrupted otherwise they will be treated as null submissions.*
- *Your work will not be printed in colour. Please ensure that any pages with colour are acceptable when printed in Black and White.*

- *You must NOT submit a paper copy of this coursework.*
- *All courseworks must be submitted as above. Under no circumstances can they be accepted by academic staff*

The University website has details of the current Coursework Regulations, including details of penalties for late submission, procedures for Extenuating Circumstances, and penalties for Assessment Offences. See http://www2.gre.ac.uk/current-students/regs

Coursework Specification

This is an individual Coursework!

You were approached by an educational company who provides educational support to children between 5 and 16 years old.

As part of their services the company decided to distribute to the pupils educational games or quizzes and they are looking to hire contractor developers to build them.

You are shortlisted to get your first job as a contractor developer for this company and to convince the company manager that you are the right person for the job, you are required to design and implement a **game or a quiz** as a standalone application using the C# programming language.

Examples of games and quizzes the company are looking for (choose ONE option):

1. **An educational quiz**

   The quiz should have between 10 and 20 questions on a particular subject such as Mathematics, English, Science or Computing. The questions should be for a particular level. Desired features:

   - there should be different types of questions (multiple answers, filling blank spaces, order options, etc.)
   - questions can have an image
   - the quiz should calculate a score at the end

   For ideas of quiz questions you can check https://www.educationquizzes.com/free-quizzes/

2. **Develop an elephant's memory**
   There are 12 cards, with animal images and 12 with the names of the animals, so in total 24 cards. At the beginning of the game the cards are randomly placed in a grid of 4X6, all face down. The player turns 2 images at the time. If the two cards consist of a matching image and name, then the cards are removed, otherwise they are paced back in the grid. The game finishes when the player removes all cards. Further ideas: the player could choose what type of cards they want bids, plants, etc., or how many pairs of cards. You could also include a timer, which keeps track of how long the user needs to complete the quiz.

3. **Spell to Score**
   The game presents randomly "floating" letters hidden between other objects. There is a little device available (gun, cannon, arrow, etc.) which can be moved using the keyboard and/or mouse. The aim of the player is to shoot the letters in a desired order (for example alphabetical or you can implement a shooting version of the hangman). The same game can be applied to numbers (for example, shoot only the squares, or ascending order, etc.).

4. **Cats Love Maths**
   Have a kitten move through a labyrinth to collect special numbers (for example, only the odd ones).The kitten would start with 3 lives and loose one for every mistake. Also, the kitten would get special powers if it collects 6 correct numbers in a row - for example it could get a new life.

5. **Travelling Broadens the … Score**
   At the beginning of the game, interesting facts about a particular country are displayed, as well as the map, etc.).
   Once started, the game shows little images “flying“ around (some images can be identical). There is a little flying device available (airplane, balloon, etc.) which can be moved using the keyboard and/or mouse. The aim of the player is to collect only the images belonging to that particular country.
   Negative points would be given for collecting wrong images.

6. **Another game or quiz of your choice**
   IMPORTANT NOTE: Your game idea must be approved by your tutor or lecturer before the 1st of November. If not, you are required to implement one of the above.

The application you build should store the player’s name and all the scores (for the game or quiz) for that particular player. The player should be able to see his/her past performance as a graph.

Deliverables:

All deliverables are mandatory and they should include technical documentation, user documentation, self-assessment, an electronic copy of the coursework application and acceptance testing.

Technical documentation

Please make sure that you include a title page and table of contents.

1. **Design diagrams** using UML. At least:
   a. use case diagram
   b. class diagram
   c. This section must also contain an explanation of your design (approx. 500 words)
2. **A list of bugs/weaknesses and/or strengths** in your system (approx. 500 words).
3. In the appendix include:
   a. **The source code** written by you, including appropriate comments**.** **Do not include automatically generated code.**
   b. **Completed self- assessment form**
   c. **User documentation with:**
      1. A concise specification of what your application does (approx. 200 words).
      2. Screen shots of the working application together with explanations.
      3. User instructions if appropriate

The electronic copy of your coursework application

An electronic copy of your coursework application should be uploaded by the coursework deadline. You will have to zip up the folder before you upload it.

NOTE: If solutions prove too big for uploading, you can delete the contents of the Bin folder from your solution before compressing the application, but do NOT remove the source code (*.cs files).

Acceptance testing

For the acceptance testing you need to demonstrate to the client (in this case your tutor) the system you have built. You will be asked to demonstrate the required business functionality.
The time and day of the acceptance testing will be made available by your tutor prior to the submission date. **Failure to attend this will result in an automatic fail for this assessment**.

Self-Assessment:

You will find a self-assessment sheet attached at the end of this coursework. You are to complete this sheet and submit it with your report. The grade that you award yourself is NOT the final grade that you will be awarded. Your coursework will still be graded by an academic member of staff. There are 5 marks allocated for accurate self-assessment.

Grading Criteria

In general the marks are awarded as follows:

- >=70% well documented work that follows the requirements both in design and implementation to a high standard and which shows detailed critical understanding of relevant concepts with an excellent report and demonstration.
- >=60% work that follows the requirements both in design and implementation well and which show very good understanding of relevant concepts with a very good report and demonstration.
- >=50% work that follows the requirements both in design and implementation reasonably well and which show good understanding of relevant concepts with a good report and demonstration.
- >=40% work displaying minimal functionality, and generally correct use of concepts with a reasonable report and demonstration.
- >=30% work displaying little or incorrect functionality, and a simplistic or partially incorrect use of concepts with a poor report and/or demonstration.
- <30% work displaying no functionality, and a simplistic and incorrect use of concepts with a very poor report and demonstration.

Assessment criteria:

- Object-oriented design: 15 marks
- Features and concepts used in your application: 70 marks
- Acceptance Testing : 10 marks
- Peer–assessment: 5 marks

Self-Assessment Sheet – on the next page

Place **a tick in the box that you deem to be most indicative of the quality of the work**.

Please include comments for each element of the desired features to justify your choices (for example, for overloading, state where you used it and how many times**).**

Note: You must submit this self-assessment together with the technical documentation. The boxes in bold are for the tutor.

Please bring a printed copy of the completed self-assessment form to the acceptance testing session and give it to your tutor.

Marks mapping:
Poor (P) = failing mark <40%;
Acceptable (A) = 40-49 %;
Sound (S)= 50-59% ;
Good/very good (G) = 60 – 69 %;
Excellent (E)= 70-80%;
Outstanding (O)= mark >80%

Self-Assessment Sheet

NOTE: You must include comments for each section of the self-assessment sheet

Student Name:______________________________________

	%	P	A	S	G	E	O	Comments
Object Oriented Design (including UML diagrams and how the design is reflected into code and any other technical documentation included in the report)	15							
Use of desired features and concepts								
• Well named variables and, methods. Decisions, Iteration. Overloading	10							
• Objects and Classes. Collections	10							
• Events controls. Validation	10							
• Threads. Animations.	10							

- Inheritance. Interfaces - Persistence – storing and reading data from a db - Patterns - Extra features (such using web services)	10 10 5 5							
Acceptance Testing (includes how well the app is presented and the ability to answer technical questions)	10							
Accurate Self- Assessment	5							
Possible AO?								