Soft152: Software Engineering 2015 - 16

COURSEWORK

There are two options for this assignment. Choose any ONE. Both have the same weighting and the same marks.

This assignment will contribute 70% towards the overall module mark

Option 1: Data Analyser

In brief the purpose of this application is to read modified response files generated from Moodle and to provide simple descriptive statistic and bar chart / pie chart representations of results.

In total there are 8 data file, representing the results from quizzes A…H

Each data file is of exactly the same format described as follows:

The first four columns are:

Index	ID	Attempt Number	Time taken
1	2021673	1	10 mins 23 secs
2	1014676	1	9 mins 2 secs
3	2815685	1	4 mins 58 secs
4	2816679	1	12 mins 54 secs
5	2021673	2	9 mins 42 secs

Where:

Index: Is just an index number to identify a particular attempt. (Not important for the data analysis, but the index represents a chronological ordering of quiz attempt: index 1 was done before index 2 and so on)

ID: Is a 7 digit unique ID number of the individual completing the quiz.

Attempt Number: Individuals may have attempted the quiz more than once. This identifies that individual's attempt number. So for example in the data above one individual (ID number = 2021673) attempted the quiz twice

Time taken: As the column heading suggests this is the time taken to complete the quiz for that particular attempt number. This time taken will be expressed in one of: just seconds; just mins; mins and seconds; hours and minutes; and in some cases days and hours. In the case of the latter (i.e. days this can be ignored for calculation purposes).

Following these four columns are further columns showing the overall result of that attempt and the score obtained on each question.

Each quiz is graded out of 10. In the sample shown there are a total of 5 questions

Overall grade	Maximum mark for each question				
10	**0.63**	**1.25**	**1.25**	**3.75**	**3.13**
8.75	0.63	0	1.25	3.75	3.13
8.13	0.63	1.25	1.25	1.88	3.13
9.38	0.63	1.25	1.25	3.13	3.13
7.5	0.63	0	1.25	3.13	2.5
10	0.63	1.25	1.25	3.75	3.13

As mentioned there are 8 data files representing results from 8 quizzes. While format is the same for each data file, the number of questions in the quiz does vary, as shown below:

Quiz	Number of questions
A, B, C	5
D	12
E	6
F	9
G	4
H	11

Thus the first 3 quizzes (A...C) have the same number of questions.

All data files are in a comma separated format (.csv)[1]

The data files can be found in the .csv data file folder in the Assignment section on the DLE. Please do not modify these files to work with your submission. When marking the original files will be used and it may result in the files not being read properly.

In addition to these .csv files there are standard Excel (.xlsx) versions to allow you to calculate averages etc. whilst validating your results. In is *not* expected nor necessary that your application is able to read these Excel files. For the excl files please see the Excel data folder also in the Assignment section on the DLE.

[1] Note a .csv file is just a text file where components of the file are separated by commas. A .csv file can be treated as a text file: Can be opened and modified in standard text editors, e.g. Notepad or WordPad.

The application you need to write should be able to read any one of these data files and provide simple text based descriptive statistics and bar chart / pie chart representations of results. The actual structure and layout of the interface will be left for you to decide. But you will need to include:

1. A file Menu with only option: File Open

2. A panel on which will display the average score of each question for all attempts as either a bar chart or a pie chart. Note the average score should be shown as a percentage of the maximum of that question. Provide either radio buttons or check boxes allowing the user to switch between the two chart types.

   [Note: When drawing the graphs, do ***not**[2] use any C# chart components. For the bars use the standard graphics methods of DrawRectangle() or FillRectangle(). Similarly for the pie chart use the graphics methods of DrawPie() or FillPie() For more details please see: https://msdn.microsoft.com/en-us/library/system.drawing.graphics%28v=vs.110%29.aspx]

3. A multi-line text box which will show in text (i.e. non graphical) format:
   i. Quiz name

   ii. Number of first attempts
   iii. Total number of attempts

   iv. Average grade of first attempts (as a percentage)
   v. Average grade of all non 1st attempts (as a percentage)

   vi. Average time (in decimal minutes i.e. 15.27 mins) to complete first attempts
   vii. Average time (in decimal minutes) to complete all non 1st attempts.

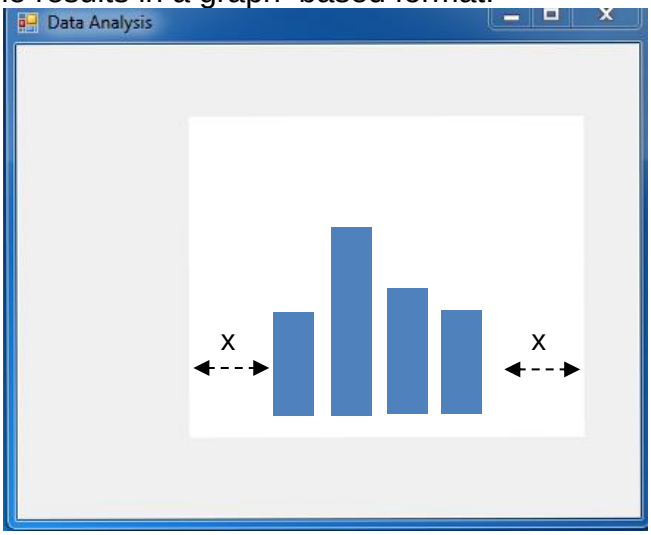
[Note: Some times are given in **days**. Ignore these instances, i.e. do not include them in the averaging]

So for example the output in the multiline text box should be similar to this:

Quiz name	Quiz A
Number of first attempts	119
Total number of all attempts	208
Average grade of first attempts	60.12%
Average grade of all attempts	68.08%
Average time to complete 1st attempt	17.87 mins
Average time to complete all non-1st attempts	15.31 mins

[2] Unfortunately if a chart component is used then marks will be deducted. Please see the marks penalty section.

When displaying the results in a graph based format:



The bars should be drawn in the centre of the panel, with an equal gap between the end bars and the panel. So for example when drawn vertically, the gap 'x' should be the same both on the left and the right. Anchor the panel on which the bars are drawn so that it expands when the form is re-sized. If done correctly then the bars should remain in the centre of the panel.

Note: There is no need to draw the characters 'x' or the arrows shown. In a similar vein the pie chart should be drawn in the centre of the panel

Notes:

1. If it is too challenging to work with all the data files where a variable number of questions needs to be coded for, then please only consider the first 3 quizzes (A...C) as these have a constant number of questions. This will result in a slightly lower mark, but not significantly so.

2. On the other hand for extra credit some of the following may be considered:

   i. For the graphical output, include controls which will allow the charts to switch between showing average results of 1st, 2nd 3rd attempts etc.

   ii. For the text based output maybe also show:
   Standard deviation of first attempts
   Standard deviation of all (non 1st) attempts (as a percentage)

   iii. Also statistics (e.g. average grade, average time) for 1st, 2nd 3rd attempts etc.

3. Do not hard code an absolute path to the folder which contains the data files. An example of an absolute path is something like:

   C:\Users\<user name>\ Documents\QuizMarks\QuizA.csv

All opening of files should be done using standard file open / File Save dialog boxes. There is a marks penalty if absolute addressing is used, please the 'Penalty for missing elements' section of this document.

Option 2: Ants collecting Food

This is not strictly a game, but simulates a possible technique which Ants may use when foraging for food. This scenario is to be developed in Visual Studio 2012 / 13 / 15. Any other environment (e.g. Unity, etc.) is not possible,

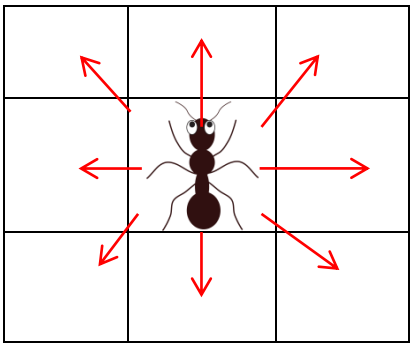
In the domain of Artifical Life (ALife) the behaviour of social insects has been used to study how simple creatures can be used to solve complex problems.

While individually insects are behaviourally unsophisticated, they collectively can solve complex problems. For example, 'real world' ants can:

- Form bridges by chaining their own bodies
- Build and protect nest
- Sort brood and food items
- Co-operate in carrying large items
- Find shortest route to food source

It is this collective 'intelligent' behaviour which has led to many studies and simulations. The following involves a group of ants collecting food and returning the food to a nest is just one possibility.

The ants live in a 2D discrete (grid based) world[3]. For ease of movement it maybe an idea to wrap the world into a toroidal shape. At each time step each the ant can move one step in one of the possible eight directions:



At the start there is no food or nest in the world and a number of ants are created at random positions in the world. These ants do not where food is, nor do they know where their nest is. Their eventual role is to find food and carry it back to a nest. The ants move randomly until they:

- **Either** encounter another ant who knows were food or a nest is, in which case they ask the ant for the information they need. What happens now depends on if they have food or not. If they have food they steer towards their nest, if they do not have food they steer towards the food.

[3] A discrete (grid based) world, must be used. Do not submit an assignment which uses a continuous based world, or one which is based on a continuous world. For either of these cases the assignment will not be accepted (i.e. will receive a mark of zero for the coding element)

- **Or** they walk close to a food or a nest.

In either case they remember the location of the food and / or the nest.

Once they deposit food at the nest they steer back to where they 'remember' where the food is.

If they arrive where they thought food was and none is there (because other ants have picked it all up), they forget were the food was and now move randomly.

Each ant will need to have a 'radius' within which they can detect food / nest, and detect other ants who may have information about food / nest. These two radii may be different.

At any given time there maybe more than one ant in the radius. You may decide to only ask the closest ant or all of them in order to find out where the food was.

Allow the user to deposit food using mouse clicks. Assume one click leaves n-units of food (the user can place several bits of food in close proximity) and each time an ant picks up food it picks up m units (also m is much less than n, i.e. m << n), so the food will eventually disappear.

Also allow the user to place nests in the world via mouse clicks.

Assume the ant is a bit forgetful: It may forget where its nest was or forget where the food was (or both!) and as outlined above will need to either stumble across it or ask another ant.

Extra credit:

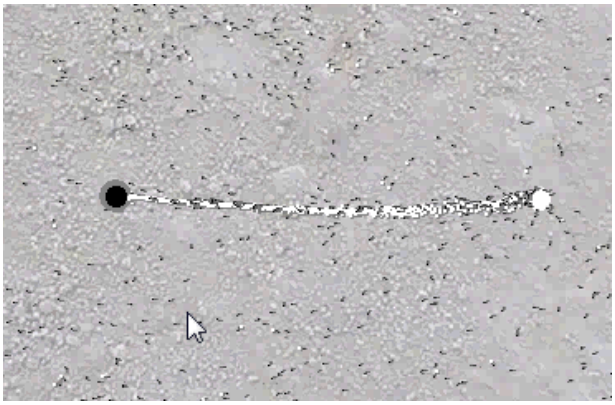
Allow for a separate population (or populations) of ants who steal the food off the other ants. These robber ants have their own nest and do not get food from the food piles, but move randomly until they find an ant with food. They can remember where this ant is. The chances are the ant with food is in a chain of ants carrying food and thus there is a good chance returning to that position

Some screen shots of an application is shown below

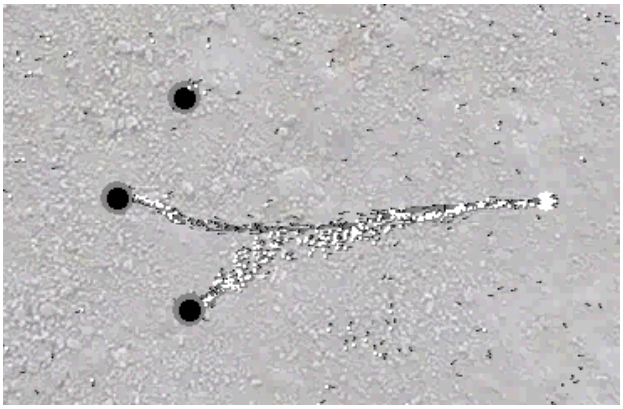
[Note the algorithm is slightly different to that given above and is based on a continuous world, but should elicit similar behaviour]

Start of simulation nest (in black) and food (in white). The ants are the tiny black dots.

Ants begin to swarm to food and carry food back to the nest



Definite path between food and nest emerging



User added another nest, now some agents choosing the new nest

A short video of this simulation is in the DLE -> Assignment section

Deliverables: There are **two** deliverables for this assignment, a report, and the actual coded solution.

1. **The Report:** The report will be in **two** sections:
   a) Describe your solution to the problem:
      a. Describe the classes used – their properties and methods include a UML diagram – annotated if needed.
      b. Description / explanation of use of any data structures used, e.g. List<T>
      c. Any enhancements or features included which were NOT in the original

   b) An evaluation section (MIN 300 words; MAX 600 words):
      a. A critical evaluation of how successful you believe your application actually is, e.g. any bugs (there is always one!), things that do not work perfectly / as hoped.

      b. With the benefit of hindsight what you may have done differently **in terms of coding or the design of classes / relationships between classes / data structures.**

2. The actual coded solution. This is the Visual Studio project containing all the C# code needed to compile and run your solution. The code should be documented (please see next page for details).

Section A of the report should contain sufficient detail to explain to a user the: structure of the solution; rationale of why approaches were taken; parameters / assumptions used.

The specific marking guide is given on the next page, but generally will be looking for what can be termed as good programming practice:

Modularity

- As a minimum an appropriate demonstration of modularity by use of methods.
- A better approach would be appropriate use of a class(es) other than the default one (i.e. the form class)

Adherence to OO principles (information hiding / encapsulation)

- Correct use of access modifiers (e.g. private, public) for both instance variables and methods.
- Appropriate use of getters / setters (e.g. not giving all instance variables both read / write access by default)

Code Complexity / readability

- Limited use of 'magic numbers'
- Preference for high cohesive methods, (e.g. those which perform a single conceptual task)
- Appropriate variable names conforming to C sharp conventions

Documenting

- Appropriate descriptive comments for all classes used
- Appropriate comments for all instance variables.
- Appropriate comments describing all methods (event handler methods can usually be excluded as can getter / setter methods)
- Inline commenting where appropriate.
- All commenting should comply with C# convention.

Marking Guide:

	Mark
Report Appropriate description UML diagram Object interaction	40
Evaluation Realistic suggestions / improvements in code / design	10
Program implementation Criteria: C# coding conventions in method / field / property names. Appropriate use of modularised code, e.g. methods and classes. Succinctness and clarity of code. Use of basic OO concepts (e.g. information hiding, appropriate use of public / private and 'getters and setters'). Use of built-in types / arrays / List<T>(where needed) Error handing / input validation	50
Total marks	100

This assignment will contribute 70% towards the overall module mark

Please note this is a core module for Computer Science / Computing and Games Development, thus if the submitted application does NOT compile it will not be considered for a pass (the mark will be capped at under 40%). If your application does not compile, then comment out whatever code is causing the error.

Assed learning outcomes:

This assignment measures all four learning outcomes: LO1, LO2, LO3, LO4.

[Please see the module overview or the 'Aims & Learning Outcomes' section on the SOFT152 DLE]

Penalty for missing elements:

Unfortunately each year a number of assignments are submitted, which either have some parts missing or require some work / time on my part to work out (or fix even!) what is being submitted. To prevent this, the following marks will be LOST according to:

Reason	Marks LOST
Missing or Visual Studio project structure (e.g. only the .cs files submitted)	20
Using hard coded absolute paths for file(s) containing details of the modules.	20
Using a chart component instead of using the built in Graphics methods	30

For this assignment you may optionally work in pairs if you wish. Or you may work and submit an individual assignment. There are no mark differences between working individually or in pairs per se.

If you decide to work in pairs then you need to stay in that pair and submit a combined effort. If you start working in a pair then decide to work individually, then each individual solution must be wholly unique. If identical or very similar sections of code / algorithm are submitted by two or more individuals as individual submissions then all those submission could be considered as plagiarism. Please see the plagiarism section on page 13 of this document. So please do consider carefully before you work as a pair.

Submission deadline**:** **09:00 Monday January 18th 2016**

Please place your report in a folder within the Visual Studio project directory. Zip that entire directory and submit **via DLE. If working as a pair then both parties need to submit their submission under their own name[4], but with the report suitably labelled (see below)**

The report should have a cover page and ideally a table of contents. The cover page should contain information to identify the authors. This should be at least student ID then, optionally, name.

If working in pairs then the identity of BOTH the authors should be on the cover page. If identical (or closely similar) work is submitted individually by two or more students, then again this can be seen as plagiarism. So please if you have worked as pair make sure both of your identities are on the same report.

Clearly if a student's ID / name are not on a report, then it is assumed they are not one of the authors. Once the deadline has passed it will not be possible to add your identity to a report.

Please make sure all files and folders are present when submitting. You may submit as many times as you wish prior to deadline if you find something was not quite right with the submission so far. Anything re-submitted will automatically overwrite anything uploaded so far. Please be advised: If files are missing or the Visual Studio solution is corrupted in some way so it is not possible to compile your solution then the mark for the implementation part of the assignment may be zero: It will NOT be possible to email them after the submission date. Suggest you zip up the Visual Studio solution folder and un-zip and try on a different machine ☺

Return of marked work: Assignment will be marked by Monday 15th February 2016

[4] Unfortunately the way DLE is set up, a non-submission is automatically identified by the system. So to show that you have completed and submitted an assignment, then both parties in a pair need to submit their submission.

Plagiarism:
As mentioned this assignment can be worked on and submitted either individually or in pairs. If completed as an **individual** assignment then the submitted work must reflect the work of ONLY that individual. Similarly if working in pairs then the submitted work must reflect the work of ONLY the two individuals in that pair.

Thus, while you may discuss, **in general terms**, this assignment with your colleagues, the assignment **MUST** be your own work.

Do not: Share designs or code with anyone, OR Submit a program design or code which is wholly or partially the work of someone else.

The University treats plagiarism very seriously. In all cases of suspected plagiarism, formal action will be taken

The penalty for submitting work which is wholly or partially the work of someone else is usually, at least, a mark of zero for the assignment. Also do not be tempted to help a colleague out who is 'stuck', by giving them your code or design, as BOTH parties will be guilty of an assessment offence and BOTH face the risk of a zero mark. Please refer to your student handbook for guidance as to what constitutes original / individual work.

The module leader may viva students on the contents of any part of their submission of their assignment. Failure to attend a viva may result in a zero mark for the assignment.

Ok so how much can you share with your colleagues? The following table gives an indication as to what is allowed / not allowed:

Allowed
Discussing in general terms, i.e. if started / finished assignment; how easy/difficult; clarifying requirements (but should email me to confirm).
Helping to fix compiler errors, but NOT suggesting a better way to do the coding.
NOT Allowed
Suggesting better way to do things, i.e. giving or suggesting an algorithm[5]
Meeting as a group to discuss design and / or algorithms and then, later, individually coding in the design / algorithm
Give ANY: code, design or algorithms to a colleague.
Using someone else's designs/ code found on some media (i.e. a hard-drive), or some printed document found somewhere
Working collaboratively with a colleague or colleagues to complete the assignment
Getting some else to complete the assignment for you!

Essentially if the level of collaboration is not covered by the allowed section, above, you MUST assume that it is NOT allowed. If you have any doubt then email the module leader to conform the level of collaboration BEFORE you do it!

[5] Note an Algorithm in the table refers also to giving or sharing pseudo-code.