Class Assignment – Hare and Tortoise
Part A

Due Date: 26 October, 2015 – No late submissions accepted. Weighting: 40%

Specification Version 1.0 (20th September 2015)

In this assignment you have the option of working in pairs. If you work alone, you will not receive any special consideration. Programming pairs need to be registered with Mike Roggenkamp by email before COB 16th October, 2015. See page 9 regarding registering pairs.

Given more than 4 weeks to complete this assignment, there will be no extensions granted under any circumstances. If illness or other exceptional circumstances prevent you from submitting your assignment by the due date, email Mike, m.roggenkamp@qut.edu.au; ***Do not submit a formal request for an extension as the Examination period commences on 31th October.***

Plan to submit before 26th October, 2015. Submission details will be available on 12th October.

1. Introduction

This assignment aims to give you a "real world experience" that occurs far too often in the workplace. You have been hired to complete a project that has not been fully specified at this stage. You have been given (in your mind) an impossible delivery date. There is some supporting documentation to the project in addition to this document. You are required to deliver a working prototype project by the required date.

This assignment is known as the **Class Assignment** because it uses a number of multiple C# classes in contrast to your earlier assignments where all of your code was in a single class.

2. The Task

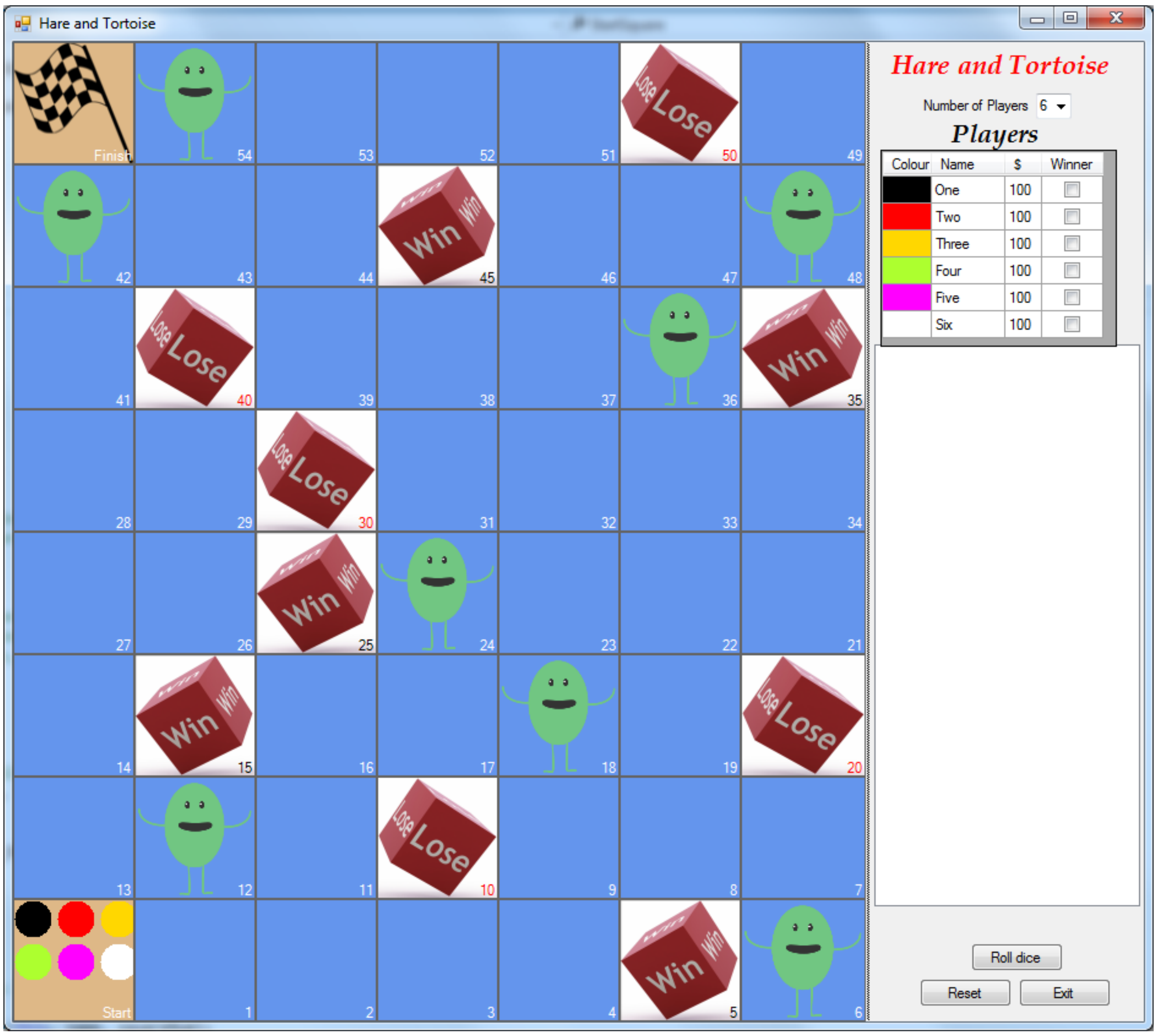
The Good Product Software Company has hired you to complete the implementation of a prototype that is at an early stage of development. It is a Board game, named **Hare and Tortoise**. The programmer who was to implement the prototype has left the company at short notice and has left behind the prototype project folder which is basically empty.

You are to implement the prototype of a board game. You are required to

(a) Use **Windows Forms**, rather than other GUI technologies such as WPF, XNA, web-pages, game engines, etc. (If you don't know what some of these acronyms mean, that's fine. Just ignore them.)

(b) Produce a GUI layout that is very similar to the screenshot on the next page

(c) Develop the game which will play the game according to the specified rules which will be explained in a later update of this assignment.



Colour	Name	$	Winner
⬛	One	100	☐
🟥	Two	100	☐
🟨	Three	100	☐
🟩	Four	100	☐
🟪	Five	100	☐
⬜	Six	100	☐

The above screenshot shows the game with 6 players, at the start of the game. Each of the 6 players has a coloured token (or "piece") on the **Start** square. (If you want to see a bigger picture in **Word**, right-click on the screenshot, select **Format Picture...,** then the **Size** tab, and then increase the **Height** – or **Width** – to 80% or more.).

3. The Hare and Tortoise Game

This is entirely fictitious board game and bears no resemblance to the fable of the Hare and the Tortoise (by Aesop) or any exsting on-line game of the same name.

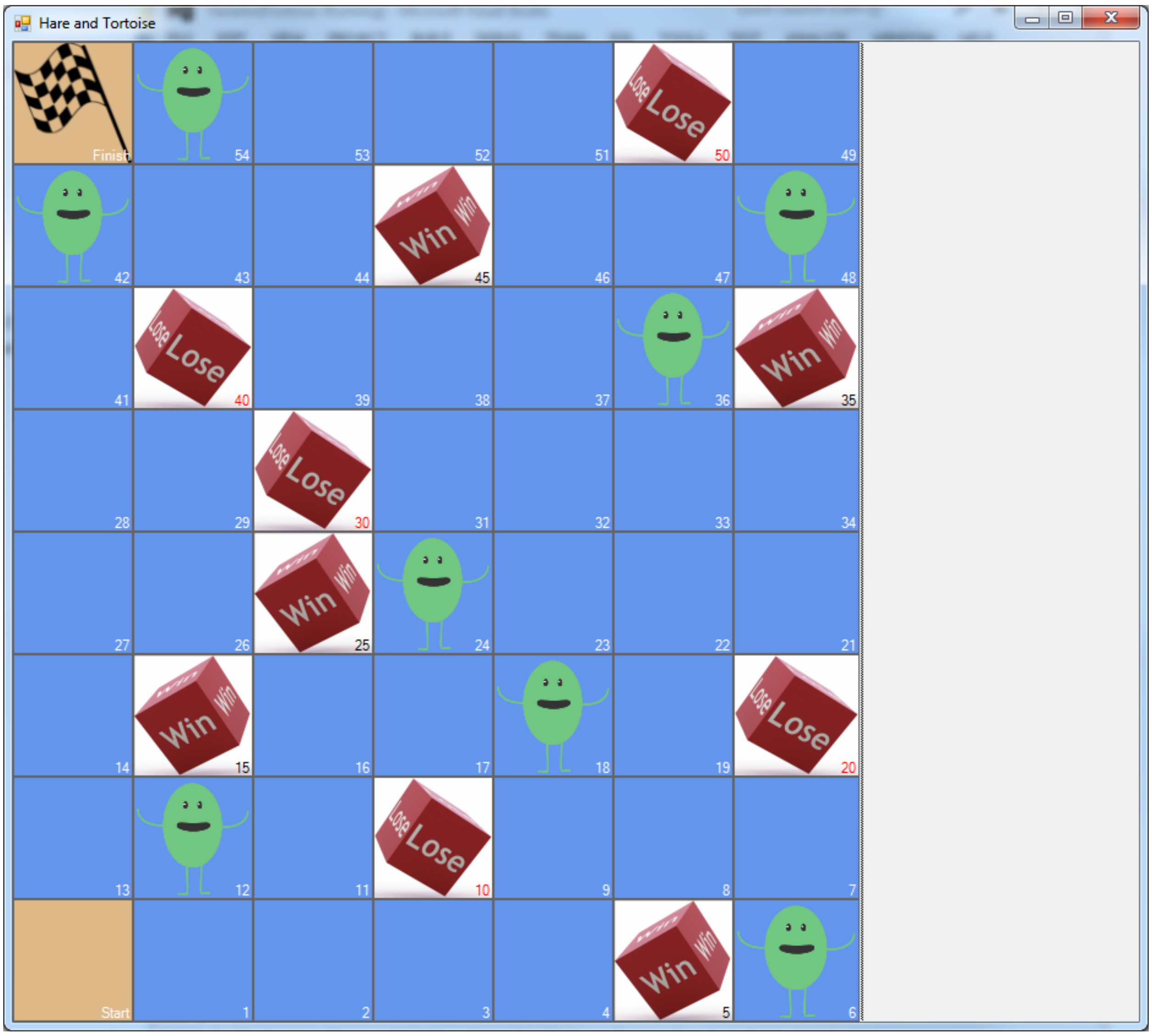
The game is played with a board consisting of 56 squares and can be played by 2 or more players up to a limit of 6 (for the purpose of this assignment only). The 56 squares consist of a Start square, a Finish square and 54 board squares, numbered 1 to 54.

There are four (4) types of board squares: "ordinary", "lose", "win" and "green monster chance". The purpose of these will be explained in later documentation.

- Squares 10, 20, 30, 40 and 50 are "Lose" squares.
- Squares 5, 15, 25, 35 and 45 are "Win" squares.
- Squares 6, 12, 18, 24, 36, 42, 48 and 54 are "Chance" squares
- All other numbered Squares are "ordinary" squares.

4. The Objective of Part A

This objective of this part of the assignment is to implement enough of the code of various classes to have the GUI form looking like



5. HareAndTortoise Project folder

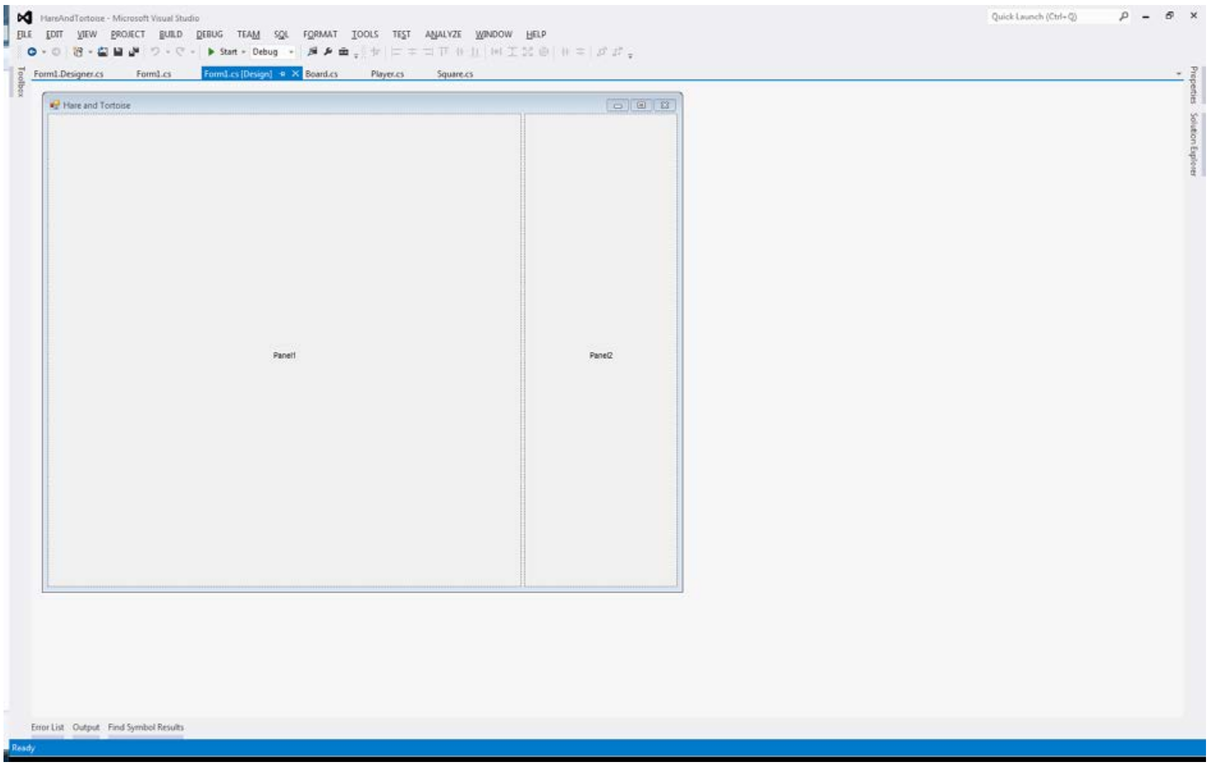
The project folder contains five (5) projects. Open the Solution file **HareAndTortoise.sln.**

In the **Solution Explorer** window you will see the projects (in alphabetical order):

- **Board Class Library**
- **Die Class Library**
- **HareAndTortoise**
- **Player Class Library**
- **Square Class Library**

The project **HareAndTortoise** contains the GUI Form, **HareAndTortoiseForm.cs**

Opening the form you will see that the form contains a ***SplitContainer*** which has divided the form into two panels: one on the left, for the game board; and one on the right for all other controls.



You can't really see the **SplitContainer**, just its two panels. The SplitContainer is docked in its parent container, so that its panels occupy the whole of the form, i.e. its Dock property = Fill. (This is the default behaviour when you add a SplitContainer to a form, so you shouldn't have to do anything. But if you accidentally change this property, your form will look a mess.) **This control is complete.** Its position on the form is correct. **Do not move it.**

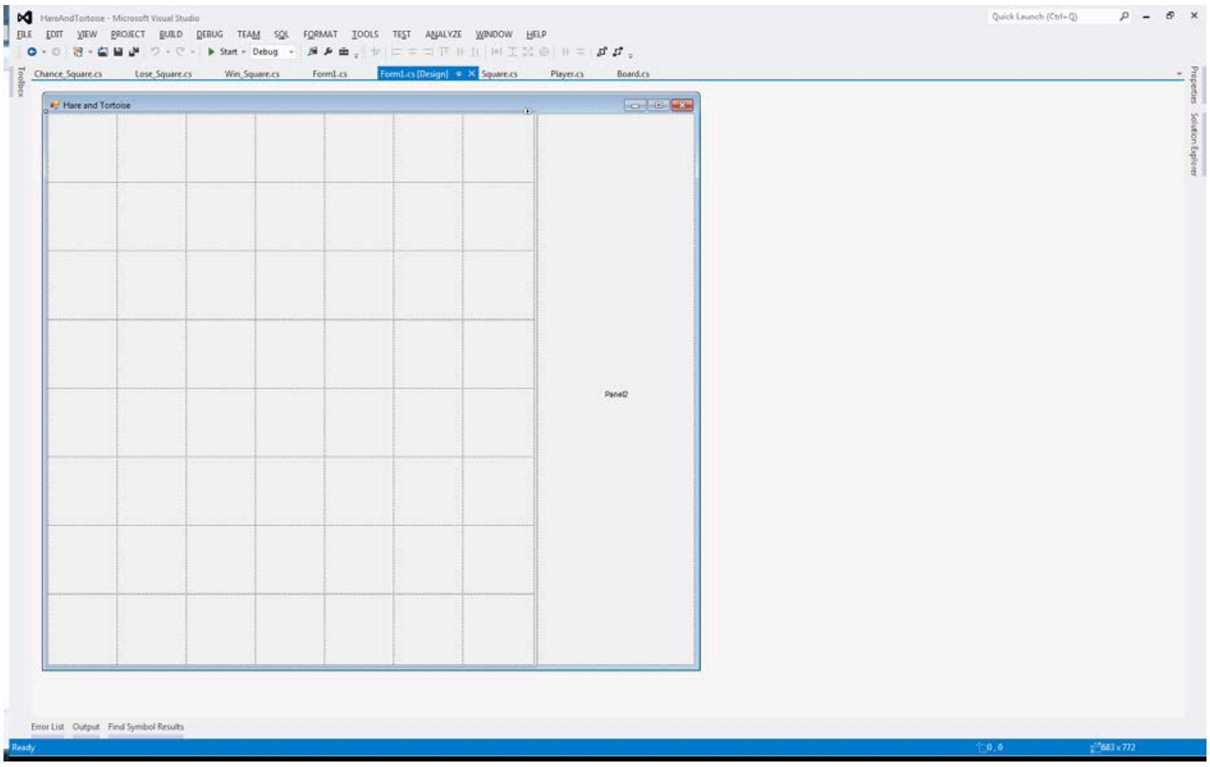
6. Start to Develop the GUI

First step is to add a *TableLayoutPanel* to the left-hand panel so that it has 8 rows and 7 columns. Rename the *TableLayoutPanel* **gameBoardPanel.** So that it always occupies the whole of the left-hand panel, you will need to set the following three properties in the order shown:

Dock = Fill
AutoSize = True
AutoSizeMode = GrowAndShrink

*N.B. set the **AutoSiz**e property **after** you set the **Dock** property or you will find that your TableLayoutPanel becomes extremely small and difficult to work with. If this happens it is easier to delete the TableLayoutPanel and add a new one.*

You form should look like this in Design View.



At this stage do not add any code to the form. The coloured squares and images will be added dynamically by code later.

7. Implement the Low Level Classes

Square Class Library

Currently contains one class, **Square.cs** which will represent an ordinary square on the board, it is also the parent (base) class for the subclasses which will represent a win square, a lose square and a chance square.

The document, **Square and subclasses UML Class Diagram**, provides UML class diagrams with some additional explanation of the classes. Implement these classes as per their specification before proceeding to implement the next class.

Board Class Library

Board.cs will represent (model) the board used to play the game. The document, **Board UML Class Diagram**, provides the complete specification for this class.

As the board consists of the various types of **Squares**, this class will need to reference the **Square Class Library.**

Player Class Library

Player.cs will represent a player in the game. The document, **Player UML Class Diagram**, provides the initial specification to be able to complete Part A of this assignment.

8. HareAndTortoise Project

This project contains four classes; **HareAndTortoise_Form**, **HareAndTortoise_Game, SquareControl a**nd **Program.**

Program.cs contains Main and no additional code is to be added to this class. It is merely the starting point for execution of the program.

SquareControl.cs is used to display (each) one of the squares on the game board. For each cell on the board, there will be a Square object and a corresponding SquareControl object from this class. This class is complete but its body is currently "commented out" to allow the program to compile and display albeit a simple form as shown on page 5 (provided you have added the TableLayoutPanel to the form as directed in section 6).

This class is a subclass of the PictureBox Control class. That is, a SquareControl object is a PictureBox. When opening the class, the Design view will show the following message

```
To add components to your class, drag them from the Toolbox and use the
Properties window to set their properties. To create methods and events for
your class, click here to switch to code view
```

Click on the link to switch to Code view. As this code uses Square and Player objects, you will need to add references to the Square Class Library and Player Class Library with appropriate directives.

You can now remove the single block comment. Code should build without errors once you have added the references and directives.

This class is complete, do not add or change any code within it. In simple terms this class creates a PictureBox which will be placed on the game board with the appropriate image when required. You are not expected to understand the code though by the end of the assignment some of you may have a better understanding of the code.

HareAndTortoise_Game.cs will contain the logic to play the game of the Hare & Tortoise. It will contain the players and associated information which will be provided in **Part B** specification document.

The body of this class is currently "commented out". Remove the block comment and add the necessary directives so that the code compiles without errors.

The players are stored in a **BindingList<T>**, which is like a **List<T>**, and it will enable a DataGridView Control to be used in the GUI. The DataGridView Control will be described in **Part B** specification document.

The class contains one method, **SetUpGame()**, which currently contains a single statement. Do not add any additional class variables or methods at this stage.

Do not start Section 9 if you have not completed Sections 6 – 8 completely.

9. Creating GUI Game Board

Open **HareAndTortoise_Form.cs** in Code View. It contains two constants, a "helper" method **ResizeGameBoard().** This method has been provided to ensure a consistent size for the cells of the TableLayoutPanel.

The Constructor contains calls to **SetUpGame()** of **HareAndTortoise_Game, ResizeGameBoard()** and **SetUpGuiGameBoard()**. Uncomment the two calls and remove the block comment around **ResizeGameBoard()**, add necessary directives so that the code will build and still display the simple form.

To complete this part of the assignment you will need to write code which will create a SquareControl object for each cell of the TableLayoutPanel. The body of **SetUpGuiGameBoard()** contains the following algorithm

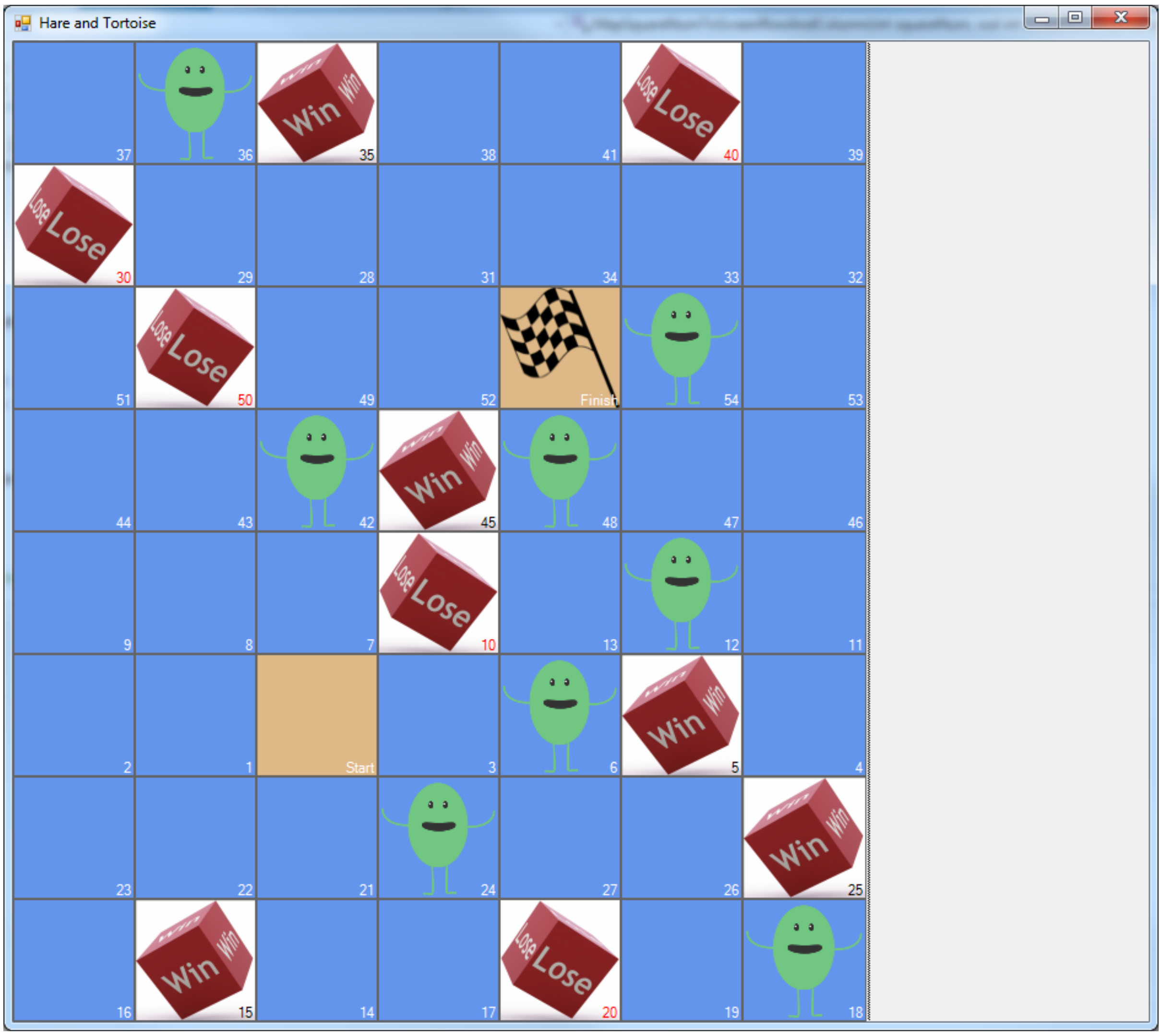
```
for each square that is on the game board
    obtain the Square object associated with the square
    create a SquareControl object (ie call Constructor)
    if the square is either Start square or Finish square
        set the BackColor of the SquareControl to BurlyWood
    otherwise do not set the BackColor
    Determine the correct position (cell) in the TableLayoutPanel of the square
    Add the SquareControl object to that position of the TableLayoutPanel
```

The top left corner of the **TableLayoutPanel** is position (0, 0) and the bottom right corner is position (7, 6). It is recommended that “*Determine the correct position (cell) in the TableLayoutPanel of the square*” be implemented as a separate method using the heading

```
private static void MapSquareToTablePanel(int number, out int row, out int column)
```

This method uses the number of the square to determine the corresponding cell (row and column) in the TableLayoutPanel.

Initially this method could set the row and column both to be 0 (zero) so that when the program runs the form will look something like



Once you can obtain a mixed up board which may differ from the image above, you can implement the code to map each SquareControl object to its correct position. The Start square at position (7, 0) and the Finish square at (0, 0) as shown in the image on page 3.

10. Final Comment

Though all care has been taken in the production of this specification and related documentation, there may be a need to notify by email any alterations/clarifications to this specification and related documentation

SO CHECK YOUR QUT EMAIL DAILY.

Working in Optional Pairs

Ensure that both people in the group are involved and are responsible for doing some part of the assignment. Do not forget to register your group details via email to Mike before COB on 16 October, 2016. Include the full name and student number of each person in your email as well as a Group name if you are creative.