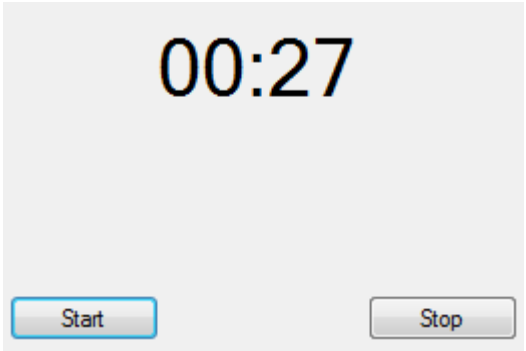
Assignment 3

Deadline:	Hand in by ***midnight Friday, 9th January 2015*** and demonstrate at the following Lab session (12th January 2015). The assignment is to be submitted via Stream.
Evaluation:	8 marks – which is 8% of your final grade.
Late Submission:	*1 mark deduction per day late*
Marking and Penalty	The assignment will be marked face-to-face with a tutor *Failure to attend the face-to-face marking session with your tutor (without prior alternative arrangement) will result in your marks being reduced by 50% (out of the possible maximum of 8 marks).*
Work	This assignment is to be done **individually** – your submission may be checked for plagiarism against other assignments and against Internet repositories. If you adapt material from the Internet acknowledge your source.
Purpose:	To learn to develop software using the TDD approach and to program event-driven applications using visual programming features of .NET

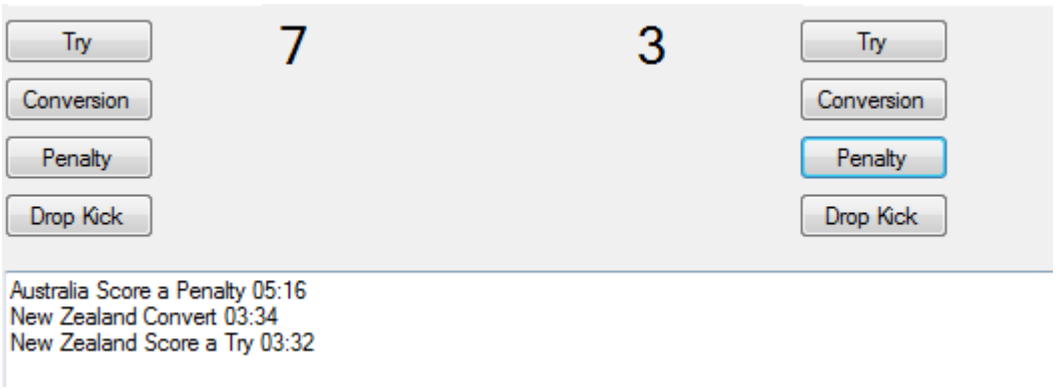
Problem:
Design an application (in C# or VB.NET) that can be used to keep score during a rugby game.

Requirements:
This application should be used to keep score during a Rugby match between two countries. At the start of the game the user should select the two countries playing and display the nations' flags. The example shown here uses RadioButtons but you can use whatever control you think is most appropriate.

Once the teams have been chosen the user should be able to start the game. The application must have a timer to keep track of the match progress. Normally a match would finish at 80 minutes but in this case it will finish whenever the user presses the stop button. The match time must be displayed in the application.



The application should also allow the user to press a button each time one team scores points. The four possible scores are a try (5 points), a conversion (2 points), a drop kick (3 points) or a penalty kick (3 points). The application should keep track of the scores for each team as well as each individual score including the team it was scored by and what time it was scored. This could be displayed like this:



When the user presses the Stop button. The game is over, the application should display which team won and what the final score is. Along with these required features, you should also think about how you are going to manage valid series of operations. A game cannot start before both teams are picked. Teams cannot score until the game has begun or after the game has stopped.

The above application must be developed using the TDD approach.

Make sure once you finish your assignment, think of one refactoring possibility (with a good reason why). You will be asked to perform the refactoring during the marking of your assignment.

Marking criteria:
Your application will be marked on functionality and the usage of good software engineering principles discussed in class, including the use of TDD approach.

Functionality (3 marks)
The application that you submit must deliver all of the required functionality. The application should anticipate, and still function appropriately under, *various types of users' behaviours* in their interaction with the application.

- Functional team selection interface (0.5 mark)
- Functional game timer with robust start and stop buttons (0.5 mark)
- Functional scoring system, including the printing of final scores at the end of the game (1 mark)
- Application of appropriate control to ensure proper sequence of operations, e.g. game cannot be started until the user selects the teams (1 mark)

TDD (3 marks)
The application should be developed using the TDD methodology. Points are awarded for:

- implementing unit tests that are appropriate, self-contained, modular, comprehensive, and independent of the main application flow control. You need not write unit tests for event-handlers but you need to write unit tests for your user-defined methods. (1.5 mark)
- using appropriate statements in the tests to ensure application robustness (1 mark)
- incorporating refactoring in the development of the code – to be demonstrated during the marking seession (0.5 mark)

Modularity (1 mark)
The application must be broken down into subroutines/functions that perform one task each. The code should be designed based-on object-oriented programming concept.

Good Coding Practice (1 mark)
Variables and function names must be named appropriately. Looping and conditional statements must follow conventional and best-practice principles. Variable scope, span, and live-time minimization. All code duplication must be eliminated. The use of appropriate line spacing and indentation to improve code readability.

Hand-in:
Submit your **program** via **stream** in a **zip** file (**ONLY** if the submission site is not available email to s.suriadi@massey.ac.nz. The assignment is due on Friday 9th January 2015. Make sure you zip your **entire** solution folder so that the .sln as well as the relevant project file are included. Good comments will help to award you marks even if your code is not quite perfect.

Assessment:
The assignment will be marked face-to-face with the tutor on 12th January 2015 during the allocated practical session time (15:00 – 17:00 at CLQB4). Please make sure you attend the practical session to have your assignment graded.

If you have any questions or concerns about this assignment, please ask the lecturer *well in advance*.