Assignment 2

Assignment 2 requires **one project** and one brief **essay**.

Project

An implementation of the **synchronous EIG-based Byzantine agreement** algorithm, working on the **K_N** graph.

The whole project must be totally contained in one **single C# source file**, called **Byz.cs**, further compilable to a **command-line executable**, called **Byz.exe**.

The executable expects one **single command-line parameter** indicating the **path** to an **input text file** describing the network.

In the submitted version, your own **standard output stream** (Console.Out) must fully conform to the sample output illustrated below. However, you can write **error messages** (if any), or any other trace messages you want, to the **standard error stream** (Console.Error).

Implementation

Use C# or F#, as available in the labs.

We suggest that you use **async** communications and **global synchronizer**, but this not a requirement.

Essay

The essay must be written as a **mock submission paper**, to a hypothetical scientific conference on distributed algorithms, with a strict page limit: **up to 5 pages** (without references). The topic is: "**Impossibility of Distributed Asynchronous Consensus with One Faulty Process**".

Suggestions: discuss why this is possible in the sync case, but not in the async case, sketch this impossibility proof, and if there ways around this theoretical limit, with what sacrifices…

Byz Input

Example:

```
Byz.exe  Generals.txt
```

where **Generals.txt** has the following content:

```
4 0
1 0 1 0 0 0 1 011 011 111 211
2 0 0
4 1 0
3 1 0
```

- Essentially, this network is the **K_4 sample** discussed in the **lectures**!
- The first input line gives the number of node, here **N=4**, and the **v_0** constant, **0** or **1**, here **0**
- Each of the succeeding **N** lines describes one **node** (participant) in the Byzantine agreement:

```
node-id  init-val  byz-faulty  [byz-faulty-script]
```

  where

  - **node-id** is a distinct index in the **1..N** range
  - **init-val** = **0** or **1**
  - **byz-faulty** = **0** or **1**, **1** indicating a **Byzantine-faulty node**
  - the **optional byz-faulty-script** indicates the messages to be sent by the Byzantine-faulty node, in the **string** format use by our **demo** (with **2** indicating null, missing or corrupt messages)
- Note that **node** lines need **NOT appear in any specific order**

Byz Output

The output has two contiguous parts: (1) a standardized trace of the **sent** messages, and (2) standardized trace of the bottom-up **evaluation**

We use a format similar to the **demo format**, further explained by the example below.

The **sent** messages are prefixed and grouped by **round** number.

The **evaluations** are prefixed by a fictive L+2 round number (here 3). We assume that the Byzantine nodes correctly make their evaluations.

Consider the preceding input scenario, where **node 1 is Byzantine faulty** and sends its messages according to the **given script**.

```
1 1 > 0 0 0 1
1 3 > 1 1 1 1
1 2 > 0 0 0 0
1 4 > 1 1 1 1
2 1 > 011 011 111 211
2 2 > 011 011 011 011
2 4 > 101 101 101 101
2 3 > 001 001 001 001
3 1 : 001000111111 0011 0
3 2 : 001000111111 0011 0
3 4 : 001000111111 0011 0
3 3 : 001100111111 0011 0
```

This format is mandatory, as the marking is largely based on exact textual comparisons.

Submit electronically, to the COMPSCI ADB dropbox, a **7z** archive of a folder containing your .CS files, a compilation batch file and your essay as PDF.

Please recheck your solutions and ensure that they compile and work in the labs (not only on your home machine)!

Please keep your receipt, and, just in case, keep also a copy of the submitted archive!

Deadline is **Monday 17 October, 2016, 18:00**. Do not leave it for the last minute, please. Remember that you can resubmit and, by default, we only consider your last submission.

Late submissions will incur penalties, 0.5% off for each hour late, for up to four days.