## Instructions:

## Computing 3: 2017/2018 Joint Assignment

## 1. Write an application to implement an application as outlined below.

## 2. The solution is to be presented in a report. This report is to contain

## a thorough description of the problem,

## a description of underlying data structures used to solve the problem,

## a copy of the code, sample input files etc..,

## a description of all the functions/routines which have been used,

## test data used to test your application, and

## a sample execution of your program on the fore mentioned test data

## 3. For Advanced programming also use your github account to store the project. Upload the source files and exe to blackboard by the deadline date.

## 4. Due Date: Feb 9th 2018 - Demos will take place on following week.

## Implement a vote-counting application for a simple electoral system.

Complete a vote-counting application for a simple electoral system based in single-seat constituencies. There are a number of candidates, one which will be elected. Each voter scores the candidates in order of preference 1, 2, 3 etc. on a ballot paper. During the count, each ballot paper is distributed initially to the can­didate who received the highest preference on that paper. Once this is done, candidates are eliminated one by one until only one remains, the last remaining candidate being deemed the winner. At each elimination the candidate with the lowest number of votes is chosen and his votes are re-distributed among the remaining candidates, with each voting paper being awarded to the remaining candidate with the highest preference on that ballot paper.

All ties are solved by random choice.

The application involves three main entities: Candidate, BallotPaper and VoteCounter that you will have to implement from scratch; their behaviours are described below. You may also need to implement other ADTs.

**Candidate** simply records the details of an individual candidate (name, party). It must support the following operations: setName, getName, setParty and getParty.

**BallotPaper** captures the idea of the ballot paper completed by one voter. It records the order in which the voter scores the candidates. We assume that there are no spoilt ballots (i.e. no errors) and that each voter will score all n candidates in order of preference from one (highest preference) to n (lowest preference).

BallotPaper support the following operations.

setPreference(n, c): Set the n-th preference for candidate c.

getPreference(n): Return the candidate who received the n-th preference on this voting paper.

**VoteCounter** simulates the vote-counting process.

This application must incorporate a mechanism that records the votes (BallotPaper) cast for each candidate by each participant. For this assignment we will assume that candidate names are unique and each candidate initially having no votes.

**VoteCounter** aim is to determine the outcome of the election based on all the votes cast and return the winning candidate.

This function should generate a detailed summary of the progress of the count at each stage as well as determining the eventual winner.