IN2029 Programming in C++
Coursework

There is a single coursework in this module, counting for 30% of the overall module mark. This coursework is due at 5pm on Sunday 24th November. As with all modules, this deadline is hard, and extensions may only be requested via the standard Extenuating Circumstances procedure.

This coursework provides practice with classes and the C++ Standard Library. It does not require any features covered after session 6.

Background

You are to implement an Alternative Vote system, a voting voting system used to select one of a small number of candidates based on preferences expressed by voters.

1. Each voter ranks all of the candidates in order of preference.

2. If the majority of voters (i.e. more than half of them) have placed a particular candidate first, that candidate is selected and the procedure is complete.

3. Otherwise, the candidate with the least number of first preferences is removed from consideration, the votes which placed that candidate first are distributed according to their next preference, and the procedure continues at step 2.

For example, consider the following collection of votes for candidates 1, 2, 3 and 4:

```
4 2 3 1
2 3 4 1
4 2 1 3
4 3 2 1
1 2 3 4
1 2 3
2 4 3
3 2
1 3 4 2
4 3 2 1
1 2 3 4
1 3 2 4
```

That is, the first voter prefers candidate 4, followed by candidate 2, and so on. Note that voters need not list all candidates. You may assume that no vote contains the same candidate twice.

Out of a total 12 votes, candidate 1 has 5 first preferences, candidate 2 has 2, candidate 3 has 1, and candidate 4 has 4. None of them has a majority, so the candidate with the least first first preferences (candidate 3) is eliminated, so the votes become:

```
4 2 1
2 4 1
4 2 1
4 2 1
1 2 4
1 2
2 4
2
1 4 2
4 2 1
1 2 4
1 2 4
```

Now candidate 1 has 5 first preferences, candidate 2 has 3, and candidate 4 has 4. No candidate has a majority yet, so now candidate 2 is eliminated, and the votes become:

```
4 1
4 1
4 1
4 1
1 4
1
4
1 4
4 1
1 4
1 4
```

There are now 11 votes, because one vote has had all its candidates eliminated. Now candidate 4 has 6 votes, which is a majority, and is declared the winner.

In this case the process terminated with only two candidates, but in other cases one candidate will achieve a majority earlier, and the process will then stop.

There is a possibility of a tie for the smallest number of votes, though this is rare with large numbers of votes. If that happens, you are free to choose any of the tying candidates.

Components of your program

You should implement and test the following classes, function and main program. Please follow the prescribed external structure and names for your classes precisely, because I will be testing your solutions automatically. You may also add private members to your classes, and any other functions you find useful. None of the specified member functions should do any output. (But you can add others that do output for testing, if you wish.)

You should divide the program into multiple files as discussed in lectures. You should use library algorithms where appropriate. Use (const) references as appropriate to avoid unnecessary copying.

You may use language features not covered in the lectures, but your code **must** be standard C++. (If using Visual Studio, set the Disable Language Extensions property under C/C++ Language to Yes.)

candidate type

Include (with the vote class described below) a definition

```
typedef unsigned int candidate;
```

We will use numbers to represent candidates, but giving a name to the type makes it easy to change this later.

class vote (30 marks)

representing the preferences of one voter.

Your class should have a single constructor

vote(const vector<candidate> &prefs) set up a vote with a sequence of candidate identifiers in preference order (first preference first).

and the following public member functions:

bool spent() const returns `true` if the vote has no preferences left.

candidate first_preference() const returns the current first preference of the vote. Clients may only call this member function if `spent()` is `false`.

void discard(candidate c) removes any occurrence of the named candidate from the vote.

class election (50 marks)

A class holding many votes.

Your class should have a default constructor and public member functions

void add_vote(const vote &v) adds a vote to the collection.

int vote_count() const returns the number of votes currently left.

void eliminate(candidate c) removes any occurrence of the named candidate from each vote, and removes any votes that are now spent.

vector<pair<candidate, int>> ranked_candidates() const returns a collection of candidates left in the election paired with the number of first preferences each has, in decreasing order of that number.[1] (If two candidates have the same number of first preferences, either order will do.)

You should also implement an external function:

election read_votes(istream &in)

that reads the votes of an election in the above format. A convenient way of extracting a line of votes is to read a line using `getline()` and then construct a `stringstream` (defined in the header <`sstream`>, which is an input stream that can be used like any other, except that it reads from the string:

```
    string line;
    ...
    stringstream s(line);
```

main() function (20 marks)

This should use the above to read election data from a file `votes.txt`. You can do this with another kind of input stream, `fstream`, defined in the system header <`fstream`>:

```
    fstream in("votes.txt");
```

Your main function should execute the alternative vote system as described above, producing output describing the process. For example, on the above sample input, it should print:

```
Round 1: 12 votes
First preferences:
  Candidate 1: 5
  Candidate 4: 4
  Candidate 2: 2
  Candidate 3: 1
Candidate 3 is eliminated.

Round 2: 12 votes
First preferences:
  Candidate 1: 5
```

[1]You should ignore the possibility that a candidate still in the election may have no first preferences at that stage, which wouldn't happen in in practice anyway.

```
  Candidate 4: 4
  Candidate 2: 3
Candidate 2 is eliminated.

Round 3: 11 votes
First preferences:
  Candidate 4: 6
  Candidate 1: 5
Candidate 4 is selected.
```

Again, you should match the specified format precisely.

Submission

Submit a ZIP file containing your source files only, to Moodle.