School of Computing & Information Technology

CSCI251 Advanced Programming
Spring 2024
Assignment 2 (Worth 14%)
Deadline: 11:55pm, 2024-09-27

Overview

This assignment is to be implemented using object-oriented programming. It involves implementing a simulation of an election campaign in a country named Verdeloria. The citizens, called Verdeloria people, are voting in an election for three political parties, from which a winner will emerge.

In this campaign, you will see ***three political parties*** and they try to win votes from ***electoral divisions***. For each party, there are a ***leader*** and some ***candidates***. Each candidate will represent the party to earn votes in an electoral division (i.e., for each party, there is only one candidate for one different division). Each candidate and electoral division will express their own's ***stances*** on ***five national issues***. The voting score received by a candidate in one division is heavily impacted by voting factors (e.g., the characteristics of the leader, the candidate and the division). In the campaign, there are several ***event days***. On each event day, each division will have a probabilistic event which impacts the voting factors. On the ***election day*** (i.e., the last day of the campaign), these factors will be finalized and used to compute candidates' voting scores. For each division, the candidate with the highest voting score (and thus the corresponding party) wins this division. The party which wins the greatest number of electoral divisions is the final winner and it will form a government. If there is more than one party with the greatest number of wins, there is a hung parliament.

Organization structure of the specification. In what follows, major components in the campaign (e.g., issues, stances, divisions, event days and election days) will be introduced in Section 1. The assignment requirement will be introduced in Section 2. The general note will be introduced in Section 3 followed by the submission guideline in Section 4.

1 Components

1.1 Issues and Stances

Issues. You need to decide on five (5) national issues (e.g., global warming, road infrastructure and rights of pets) with some simple description of why each issue is important.

Stances. For each issue, we model a stance on an issue as a tuple of 2 elements:

> The first element: the significance of the issue (quantified as a real number between (0, 1]).
> The second element: the strength of the measure for addressing this issue (quantified as a real number between (0, 1]. You do not need to specify what the measure is.)

Note: Each candidate and division will have a stance on an issue respectively.

1.2 Political parties and candidates

Each political party consists of:

1. One leader (not a candidate) with attributes: popularity (a real number between (0, 1]) .
2. One candidate (for each electoral division) with attributes: stances on five issues and popularity (a real number between (0, 1]) .

Stance's initialization: for each party, candidates should have the same *initial* stance for a specific issue. The initialization should be randomized, and you define the randomness (e.g., the sampling distribution and the sampling range). Note that these stances may change across event days.

1.3 Electoral divisions

In each division, there is only one stance for one issue. Each stance is initialized randomly, and you define the randomness. Note that these stances may be changed across event days. In addition, each division has a population attribute which is a real number uniformly sampled from [0.5, 1.5]. The unit is million and can be omitted during necessary computation.

1.4 Campaign Event Days

Each event day of the campaign involves the candidates and leaders attempting to convince the electoral divisions to support them. ***On each event day of the campaign, in each electoral division, one local event below will happen with a probability of 0.8. It means that, nothing will happen with the probability of 0.2, and the sum of probabilities of the four local events below is 0.8. It is up to you to decide how the probability of 0.8 is distributed among these events (the probability of each event needs to be a non-zero value):***

1. Two candidate-related events. One event will impact candidates' popularity and another event will affect the stances of candidates in this division. You define the rules of events.
2. One leader-related events. It will affect the popularity of the corresponding leaders. You define the rules of this event.
3. One Issues-related event. This event will affect the stances of the electoral division on the issues. You define the rules of this event.

Note: these events will not make all related attributes' values exceed their predefined ranges (e.g., (0, 1] is the range for popularity and stances). ***You need to ensure some uncertainty in all events such that we will not always have the same result if we run the same local event multiple times.***

1.5 Election day and wrapping up…

On the election day, you need to do three tasks: you need to compute voting scores and decide the winner. For each candidate, he or she will only receive one score from one division. In each division, there are multiple candidates from different parties. The candidate with the ***highest voting score*** will win the electoral division election**.**

Below is how you compute the score for one specific candidate in one division. The formula considers the following factors, listed in order of their significance.

1. The Stance-and-population Factor. The value of this factor depends on the division population and the average cosine similarity between the candidate's stances and the division's stances. For each

issue, the stance cosine similarity between a candidate and the stance in the electoral division is calculated based on their two-dimensional stances. Give two vectors/stances X=(x_1, x_2) and Y=(y_1,y_2), below is how to compute the cosine similarity:

$$\text{Cosine Similarity} = \frac{x_1y_1 + x_2y_2}{\sqrt{x_1^2 + x_2^2} \cdot \sqrt{y_1^2 + y_2^2}}.$$

The final Stance-and-population Factor can be computed as (* means 'times'):

Average cosine similarity (over 5 stances) * population in this division (a real number from [0.5,1.5])

2. Candidate popularity.
3. Leader popularity.

Therefore, the voting score of one candidate can be computed as:

A*(The Stance-and-population Factor) + B*(candidate popularity) + C*(leader popularity).

Requirements of Coefficients (A, B and C): their sum should be equal to 1 and A>B>C. You decide the values of A, B and C.

2 Assignment requirements

You need to write a program and the requirements are specified in Section 2.1. You also need to write a report in PDF format and the requirements are specified in Section 2.2.

2.1 Program requirements

Your code must compile on Capa (i.e. it must be C++17 compliant) according to instructions you provide in a Readme file. Once your program is compiled into the executable APE, it must run as follows:

./APE n m

where n is the number of electoral divisions in the nation, and m is the number of campaign days before the election. The value of n should be in the range 1 to 10 inclusive. The value of m should be in the range 1 to 30 inclusive. **You can use some containers to pre-store the names of parties, divisions and people, and retrieve these values to create objects/instances.**

Once you run the program, the program needs to output something below to the terminal:

1. Before campaign event days, you should output: party report: info of parties (i.e., all entities with attributes in each party) and (2) nation report: each electoral division with stance distributions and population.

2. During the simulation of each event day, you should output: (1) event related report: what has happened in each electoral division, and the significant impacts that have occurred, (2) party report: info of parties (i.e., all entities with attributes in each party), (3) nation report: each electoral division with stance distributions and population.

3. On the election day, you should output: a national summary, showing the number of electoral divisions won by each party. The party which wins the greatest number of electoral divisions is the final winner and it will form a government. You need to report the party leader as the new leader for the country. If there is more than one party with the greatest number of wins, you should report a hung parliament.

2.2 Content of the PDF report

The report is worth around 40% of the total score and the diagram is worth around 50% of the report. The report should have six sections. The first five sections correspond to Sections 1.1 to 1.5. In these sections, you describe the details of these elements in the program:

- For Section 1.1, describe how do you define issues with some simple description of why each issue is important, and how you model stances.
- For Section 1.2, describe characteristics of all parties' leader and candidates, and stances' initialization.
- For Section 1.3, describe divisions' names, stances' initialization and populations.
- For Section 1.4, describe events' rules and probabilities.
- For Section 1.5, describe how you derive the coefficients.

In the last section (Section 2), you need to draw the UML class diagram for this program. In the assignment folder, we provide instructions of using UMLet to draw the class diagram. You are also allowed to use other tools (e.g., draw.io and Lucid chart) for drawing the class diagram.

3 General notes

These are some general rules about what you should and should not do.

1. Your assignment should be sensibly organised with the same kind of expectations as assignment one, although you may reasonably have more files for this assignment.
2. **Other than the initial command line input, the program should run without user input. This means there should not be pauses waiting for the user to press a key.**
3. Please do not submit project, object or .txt files. You must submit only the source code files (.cpp, .hpp) necessary to generate an executable file, APE, and the PDF report.
4. **You must use classes, and should make use of encapsulation**.
5. **You must use inheritance. If you do not use inheritance, you will lose few marks.**
6. You can use polymorphism, if so, justify its appropriateness.

4 Submission Guideline

Please submit a zip file via Moodle. It contains your source (i.e. .cpp and .hpp) files, a Readme file with instructions for compiling, and the PDF report. Make sure your report is in pdf format and has your name and student number marked clearly on it. The UML-like diagram should be an integral part of your final pdf-formatted report and not in a separate document. Your report may be ignored if it is a non-pdf format.

1. Late submissions will be marked with a 25% deduction for each day, including weekends.
2. Submissions more than four days late will not receive marks, unless an extension has been granted.
3. If you need an extension, apply through SOLS before the assignment deadline.
4. Academic misconduct is treated seriously. Specifically, any plagiarised work will be awarded a zero mark and reported to the University.