Technical Software Development Assignment 1

Predicting Root Cause of Overhead Line Tripping

Submission Required: Submit soft copies of the .cpp file and report document (doc / pdf) on CANVAS http://www.swinburne.edu.my/canvas

Due Date: 5.00pm, Friday 2 April 2021

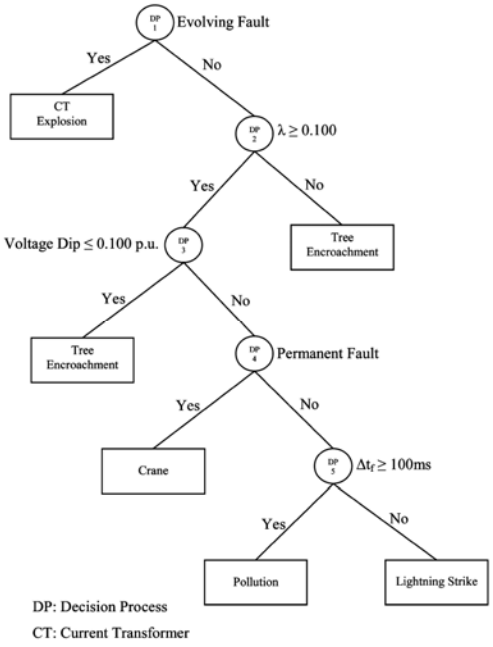

Figure 1: Decision Tree for Fault Analysis in Overhead Line Tripping in Malaysia
(Source and Image Credit: https://doi.org/10.1109/TPWRD.2007.905460)

Background

Overhead line tripping is a major cause of power outage. Researchers have developed the decision tree in Figure 1 to aid the fault analysis in overhead line tripping in Malaysia (Zin and Karim, 2007). This decision tree uses the following user inputs to predict the likely root cause of overhead line tripping:

1. Evolving Fault (binary value: yes or no)
2. λ, Gradient or rate of change of the curve (real number)
3. Voltage Dip (real number)
4. Permanent Fault (binary value: yes or no)
5. Δt_f,Time interval between the last neutral current distortions and a flashover (real number)

From these user inputs, the decision tree in Figure 1 can predict one of the followings as the likely root cause of overhead line tripping:

a) Current Transformer Explosion
b) Tree Encroachment
c) Crane
d) Pollution
e) Lightning Strike

In this assignment, you are required to develop a C++ program that obtains the input values from the users (user input 1 to 5 listed above) and predict the root cause of overhead line tripping based on the decision tree presented in Figure 1. Knowledge of variables, data types, stream input/output, selection, repetition and assignment in C++ is assessed in this assignment.

Reference:

A. A. M. Zin and S. P. A. Karim, "The Application of Fault Signature Analysis in Tenaga Nasional Berhad Malaysia," in IEEE Transactions on Power Delivery, vol. 22, no. 4, pp. 2047-2056, Oct. 2007, doi: 10.1109/TPWRD.2007.905460. DOI: https://doi.org/10.1101/2020.04.08.20057794

Assignment Tasks:

1. Software Development Task(s)
    a. Compulsory Task: Develop and submit an original C++ implementation that accepts user inputs and use the decision tree in Figure 1 to determine and display the predicted root cause of overhead line tripping.
    b. Non-compulsory Challenge Task 1: At the end of the above Compulsory Task, allow the user to choose (yes or no) to repeat the process for another set of inputs.
    c. Non-compulsory Challenge Task 2: Determine the appropriate valid value or range for at least one binary value and at least one real number input variables and use appropriate loops to validate user inputs for the variables (Hint: refer to Chapter 4 for validation loop).
2. Reporting Tasks:
    a. List the variables of your program in form of data dictionary (Hint: refer to Lab 3, Question 9(E) for example). Use appropriate data type and identifier (and style) for each variable.
    b. State how user inputs were obtained in your program. Did you obtain all the input values from the user first before entering the decision tree, or obtain the input values progressively when they are needed in each node in the decision tree? Justify your choice.
3. Testing Tasks:
    a. Prepare at least 6 screenshots with 6 sets of user inputs, one each to reach the leaves (rectangle boxes) in the decision tree in Figure 1. For each set of user input, capture the screenshot of full program input output using Alt+PrtScr (or other print screen

tool) to copy the screen image (make sure the image is clear), OR Select and use ENTER to copy the output text.

b. Prepare additional two screenshots to demonstrate Challenge Tasks 1 and/or 2 if you attempt one or both of these tasks.

Submission:

You will only need to submit 2 files through CANVAS before the due date/time:

1. The .cpp file for the Software Development Task(s). The name of your .cpp must be your student number. Do not include the Visual Studio solution files or folders or exe files.
2. The report as .doc/docx/pdf file for the Reporting Task and Testing Task AND should contain:

   - Descriptions of inputs of your program in form of data dictionary (refer to Lab 3, Question 9(E) for example) and a short discussion required in the Reporting Task.
   - Screen shots of the working program as outlined in Testing Tasks section.

Mark Distribution:

Software Development Tasks (3 marks)

Criteria	Marks	
Correctly implemented the Compulsory Task?	Yes (1 mark)	No (0 mark)
Correctly implemented the Non-Compulsory Challenge Task 1?	Yes (0.5 mark)	No (0 mark)
Correctly implemented the Non-Compulsory Challenge Task 2?	Yes (1 mark)	No (0 mark)
Good programming styles: (1) Provide header comments that contain name, date, and program description. (2) Provide inline comments to explain your code. (3) Use consistent naming conventions and indentation.	0.5 mark for implementing all three good programming styles 0.25 mark for implementing two out of the three good programming styles. 0 mark for implementing less than two out of the three good programming styles	

Reporting Tasks (1 mark)

Criteria	Marks	
Reporting Task a. Complete data dictionary AND Appropriate data type and identifier (and style) for each variable.	Yes (0.5 mark)	No (0 mark)
Reporting Task b. Statement on how user inputs were obtained in your program AND Valid justification	Yes (0.5 mark)	No (0 mark)

Testing Tasks (1 mark)

Prepare at least 6 screenshots with 6 set of user inputs, one each to reach the most bottom nodes / leaves (rectangle boxes) in the decision tree in Figure 1.

Criteria	Marks
Testing Task a. 6 screenshots with 6 sets of user inputs, one each to reach the leaves (rectangle boxes) in the decision tree in Figure 1.	0.1 mark for each screenshot of program test run correctly reaching a leave (rectangle box) in the decision tree in Figure 1.
Testing Task b. Additional two screenshots to demonstrate Challenge Tasks 1 and/or 2 if you attempt one or both of these tasks.	0.2 mark for each screenshot of program test run correctly demonstrating Challenge Tasks 1 and 2 respectively.