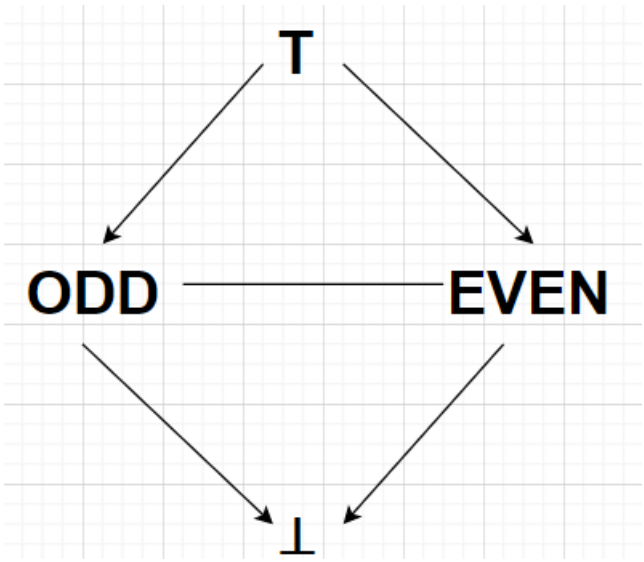
Lab 4

Q1



Q2

I would need to overwrite the following methods:

- newInitialFlow(): it is used to create a new empty flow object.
- Copy(A source, A dest): it is used to copy the contents of the source to the dest flow object, to duplicate flow information.
- Merge(A in1, A in2, A out): combining the incoming flow into one outgoing flow, used at control flow merge points where paths converge.
- flowthrough(A in, N d, A out): it defines how the analysis information changes when flowing through a specific node d. It takes the incoming flow in, the node d, and results in the outgoing flow out.

Q3

For this OddEven analysis, N would be a Soot Unit that represents a statement in the code that is being analysed. A would be the abstract state of the odd/even analysis, could be implemented as an enum with values: TOP, Odd, Even, Bottom.

Q4

OddEvenAnalysis_code.txt

OddEvenAnalysis_code.txt

This class extends ForwardFlowAnalysis to analyse odd/even state of values in a given control flow graph.

- OddEvenState enum represents the state of the value with one of 4 possible states (top, odd, even, bottom).
- The constructor initialises the analysis with the provided directed graph and triggers the analysis process by calling doAnalysis().
- newInitialFlow() returns an empty HashMap to represent the initial flow of variable states before the assignments.
- Method copy() clears the destination map and copies the state from the source map to it.
- Merge() merges two incoming states into an output state by taking the union of the variables and calculating their combined states using meetStates().
- meetStates() determines the resulting state of two states based on the rules (if both states are the same, return that state / if either is BOTTOM, return BOTTOM / otherwise, return TOP).
- flowthrough() determines odd/even state based on the right-hand side of the assignment.
- handleBinaryOperation() handles addition, substraction, and multiplication, and updates the state based on the states of the operands involved.
- getState() retrieves the odd/even state of a value (IntConstant or a Local).
- addStates() determines the states resulting from adding two states.
- multiplyStates() determines the states resulting from multiplying two states.

Q5

Main_code.txt

q5_output.txt

Q6

ReachingDefinitionAnalysis_code.txt

q6_transform.txt

q6_output.txt