#### ****1. Constructor:**** Simplex(vector<vector<double>> &A, vector<double> &b, vector<double> &c)

* **Purpose**: Initializes the simplex tableau using the input coefficients of constraints (A), right-hand side values (b), and the objective function coefficients (c).
* **Parameters**:
  + A: Coefficients of constraint equations.
  + b: Right-hand side values for the constraints.
  + c: Coefficients of the objective function.
* **How it Works**:
  + Constructs the initial tableau.
  + Adds slack variables to transform inequalities into equations.
  + Sets up the last row for the objective function.

#### ****2.**** void printTableau()

* **Purpose**: Prints the current simplex tableau along with the Zj row and Cj - Zj row to track the computation process.
* **How it Works**:
  + Prints the entire tableau in a formatted way.
  + Computes the Zj row (the sum of basic variable contributions).
  + Computes Cj - Zj row to check for optimality conditions.
  + Displays both rows.

#### ****3.**** bool isOptimal()

* **Purpose**: Checks whether the current tableau represents an optimal solution.
* **How it Works**:
  + Iterates over the objective row (last row of the tableau).
  + If any coefficient is negative, the solution is not optimal and further iterations are required.
  + If all values are non-negative, the algorithm terminates.

#### ****4.**** void pivot(int pivotRow, int pivotCol)

* **Purpose**: Performs the pivoting operation to bring a new variable into the basis.
* **Parameters**:
  + pivotRow: Index of the row containing the pivot element.
  + pivotCol: Index of the column containing the pivot element.
* **How it Works**:
  + Divides the pivot row by the pivot element to make the pivot element 1.
  + Updates all other rows to make the pivot column entries 0, ensuring a valid basic feasible solution.
  + Uses the row operations of the Simplex method to maintain feasibility.

#### ****5.**** void solve()

* **Purpose**: Iteratively applies the Simplex algorithm to reach an optimal solution.
* **How it Works**:
  + Repeatedly checks for optimality using isOptimal().
  + Identifies the pivot column (most negative value in the objective row).
  + Identifies the pivot row using the minimum ratio test (smallest positive ratio of RHS to the pivot column value).
  + If no valid pivot row exists, the problem is **unbounded**.
  + Performs the pivot operation.
  + Prints the updated tableau after each iteration.
  + Stops once an optimal solution is reached.