#include <iostream>

#include <vector>

#include <algorithm>

#include <iomanip>

#include <limits>

// Function to find the pivot column (most negative coefficient in objective function)

int findPivotColumn(const std::vector<std::vector<double>>& tableau) {

int pivotCol = 1; // Start from 1 to skip the constant term

double minValue = 0;

for (int j = 1; j < tableau[0].size(); j++) {

if (tableau[0][j] < minValue) {

minValue = tableau[0][j];

pivotCol = j;

}

}

return pivotCol;

}

// Function to find the pivot row (minimum ratio test)

int findPivotRow(const std::vector<std::vector<double>>& tableau, int pivotCol) {

int pivotRow = 0;

double minRatio = std::numeric\_limits<double>::max();

for (int i = 1; i < tableau.size(); i++) {

if (tableau[i][pivotCol] > 0) {

double ratio = tableau[i][0] / tableau[i][pivotCol];

if (ratio < minRatio) {

minRatio = ratio;

pivotRow = i;

}

}

}

return pivotRow;

}

// Function to perform the pivot operation

void pivot(std::vector<std::vector<double>>& tableau, int pivotRow, int pivotCol) {

double pivotValue = tableau[pivotRow][pivotCol];

for (int j = 0; j < tableau[pivotRow].size(); j++) {

tableau[pivotRow][j] /= pivotValue;

}

for (int i = 0; i < tableau.size(); i++) {

if (i != pivotRow) {

double factor = tableau[i][pivotCol];

for (int j = 0; j < tableau[i].size(); j++) {

tableau[i][j] -= factor \* tableau[pivotRow][j];

}

}

}

}

// Function to check if the solution is optimal

bool isOptimal(const std::vector<std::vector<double>>& tableau) {

for (int j = 1; j < tableau[0].size(); j++) {

if (tableau[0][j] < 0) {

return false;

}

}

return true;

}

// Function to print the tableau

void printTableau(const std::vector<std::vector<double>>& tableau) {

for (const auto& row : tableau) {

for (double val : row) {

std::cout << std::setw(10) << std::setprecision(2) << val;

}

std::cout << std::endl;

}

std::cout << std::endl;

}

// Function to extract and display the solution

void displaySolution(const std::vector<std::vector<double>>& tableau) {

std::cout << "Solution:" << std::endl;

for (int j = 1; j < tableau[0].size() - 1; j++) {

bool found = false;

for (int i = 1; i < tableau.size(); i++) {

if (tableau[i][j] == 1) {

std::cout << "x" << j - 1 << " = " << tableau[i][tableau[i].size() - 1] << std::endl;

found = true;

break;

}

}

if (!found) {

std::cout << "x" << j - 1 << " = 0" << std::endl;

}

}

std::cout << "Optimal Value (z) = " << -tableau[0][0] << std::endl;

}

int main() {

// Example tableau for a problem with 10 variables (x1 to x10)

std::vector<std::vector<double>> tableau = {

{-1, -3, -2, -4, -1, -5, -2, -3, -1, -6, -7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, // Objective function: -z - 3x1 - 2x2 - 4x3 ... - 7x10

{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 10}, // Constraint 1: x1 + x2 + ... + x10 + s1 = 10

{1, 2, 3, 2, 1, 4, 2, 3, 1, 5, 6, 0, 1, 0, 0, 0, 0, 0, 0, 0, 20}, // Constraint 2: x1 + 2x2 + 3x3 ... + 6x10 + s2 = 20

// Additional constraints can be added here

};

std::cout << "Initial Tableau:" << std::endl;

printTableau(tableau);

while (!isOptimal(tableau)) {

int pivotCol = findPivotColumn(tableau);

int pivotRow = findPivotRow(tableau, pivotCol);

pivot(tableau, pivotRow, pivotCol);

std::cout << "After pivoting on [" << pivotRow << "," << pivotCol << "]:" << std::endl;

printTableau(tableau);

}

std::cout << "Optimal Solution Found:" << std::endl;

printTableau(tableau);

displaySolution(tableau);