Report 1

Team information.

- Team leader: Oleynik Maxim
- Team member 1: Lobov Gleb 5/5
- Team member 2: Fakhrutdinov Bulat 5/5
- Team member 3: Kachmazov Alexander 5/5
- Team member 4: Oleynik Maxim 5/5

  All members have made maximum participation

Link to the product.

- The product is available: https://github.com/Dart-NEW/IntroToOptimizationHW1

Programming language.

- Programming language: Python

Linear programming problem.

- Maximization or Minimization?
  Maximization
- Objective function:
  $z = 5x_1 + 4x_2$
- Constraint functions:

$$\begin{aligned} 6x_1 + 4x_2 &\leq 24, \\ x_1 + 2x_2 &\leq 6, \\ -x_1 + x_2 &\leq 1, \\ x_2 &\leq 2, \\ x_1 &\geq 0, \\ x_2 &\geq 0. \end{aligned}$$

Input

The input contains:

- A vector of coefficients of objective function - C.
- A matrix of coefficients of constraint function - A.
- A vector of right-hand side numbers - b.
- The approximation accuracy ϵ.

Output/Results

The output contains:

- The string ”The method is not applicable!”
  or

- A vector of decision variables - X^*.
- Maximum (minimum) value of the objective function.

Code

Listing 1: - Python

```
def not_applicable(n):
    print(”The method is not applicable!”, n)
    exit()

def row_operation(pivot_row, pivot_col):
    matrix[pivot_row] = [f”x{pivot_col}”] + [item / matrix[pivot_row][pivot_col
    for i in range(len(matrix)):
        if i != pivot_row:
            factor = -matrix[i][pivot_col] / matrix[pivot_row][pivot_col]
            for j in range(1, len(matrix[i])):
                matrix[i][j] = matrix[pivot_row][j] * factor + matrix[i][j]

def check_unboundedness(column):
    for row in range(len(matrix)):
        if matrix[row][column] > 0:
            return True
    not_applicable(1)

n = int(input(”Enter number of coefficients of objective function: ”))
print(f”Enter coefficients (one in each of the {n} lines):”)
ls = [-float(input()) for _ in range(n)]

rows = int(input(”Enter number of rows of matrix: ”))
cols = int(input(”Enter number of columns of matrix: ”))
if cols != n:
    not_applicable(2)
print(f”Enter matrix {rows}  {cols} (one row of the matrix in each of the {rows
matrix = [[0 for i in range(n + rows + 2)] for j in range(rows + 1)]

matrix[0][1:n + 1] = ls
for i in range(rows):
    row = list(map(float, input().split()))
    matrix[i + 1][1:cols + 1] = row if cols == len(row) else not_applicable(3)
    matrix[i + 1][0] = f”s{i + 1}”
    matrix[i + 1][n + i + 1] = 1

print(f”Enter right-hand side numbers (one in each of the {rows} lines):”)
for i in range(rows):
    rhs_i = float(input())
    if rhs_i >= 0:
        matrix[i + 1][-1] = rhs_i
    else:
        not_applicable(4)

accuracy = int(input(”Enter approximation accuracy: ”))

while True:
    min = 0
    pivot_row = 0
    pivot_col = 0
    for i in range(len(matrix[0])):
        if i > 0:
            check_unboundedness(i)
        if matrix[0][i] < min:
            min = matrix[0][i]
            pivot_col = i
    if min != 0:
        min_positive = float('inf')
        for i in range(1, rows + 1):
            try:
                ratio = matrix[i][-1] / matrix[i][pivot_col]
            except ZeroDivisionError:
                ratio = -1
            if min_positive > ratio > 0:
                min_positive = ratio
                pivot_row = i

        row_operation(pivot_row, pivot_col)
    else:
        break
print()

print("Decision variables:")
for i in range(1, len(matrix)):
    if matrix[i][0][0] == "x":
        print(f"{matrix[i][0]}: {round(matrix[i][-1], accuracy)}")

print()
print("Maximum value:", round(matrix[0][-1], accuracy))
```