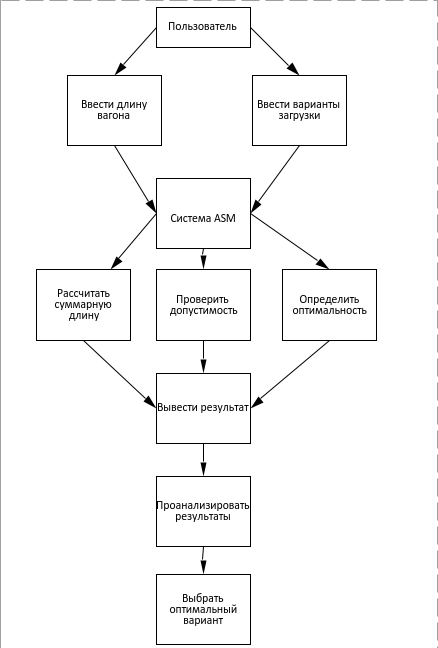
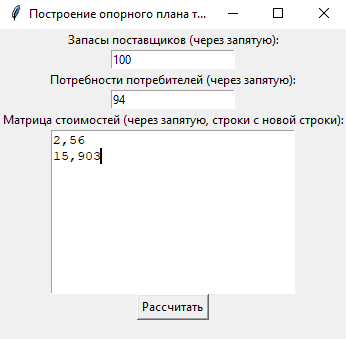
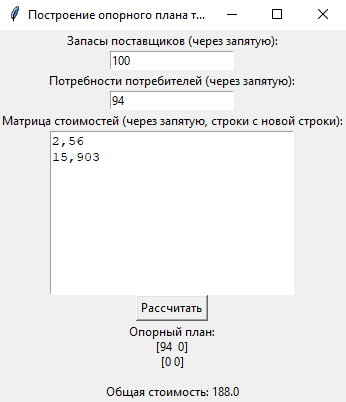
**Задание №1**

Асмо графический редактор



**Задание №2**

  
(Входные данные)



(Результат)

**Задание №3**

| **Идентификатор тестового случая** | **Описание** | **Входной сигнал питания** | **Спрос На вводимые данные** | **Входные затраты** | **Ожидаемый результат** | **Пройти /Потерпеть неудачу** | **Примечания** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TC\_01** | Основные Достоверные данные | 100,200,150 | 180,120,150 | 5,8,4\n3,7,9\n6,2,5 | Показана правильная матрица распределения и общая стоимость. |  | Стандартный рабочий кейс. |
| **TC\_02** | Неравный спрос/предложение (Избыточное предложение) | 200,200 | 100,150 | 2,5\n3,1 | Скорректируйте матрицу распределения с учетом того, что осталось несколько поставщиков. |  | Тесты на несбалансированные проблемы. |
| **TC\_03** | Неравный спрос /предложение (Избыточный спрос) | 100,150 | 200,200 | 2,5\n3,1 | Правильная матрица распределения и общая стоимость. |  | Тесты на несбалансированные проблемы. |
| **TC\_04** | Большие числа | 1000,2000,1500 | 1800,1200,1500 | 5,8,4\n3,7,9\n6,2,5 | Правильная матрица распределения и общая стоимость. |  | Проверяет масштабирование с помощью больших чисел. |
| **TC\_05** | Небольшие числа | 1,2,3 | 2,3,1 | 2,5,8\n4,1,3\n7,9,2 | Правильная матрица распределения и общая стоимость. |  | Проверяет с очень маленькими значениями. |
| **TC\_06** | Десятичные числа | 100.5, 200.75, 150 | 180.2, 120, 150.33 | 5.1,8.2,4\n3,7,9.3\n6.4,2.5,5 | Правильная матрица распределения и общая стоимость. |  | Проверяет значения с десятичными дробями. |
| **TC\_07** | Единый Поставщик и Потребитель | 500 | 500 | 10 | Правильная матрица распределения и общая стоимость. |  | Простейший случай с одним поставщиком и потребителем |
| **TC\_08** | Нулевые значения спроса/ предложения | 0,100,200 | 0,150,100 | 5,8,4\n3,7,9\n6,2,5 | Правильная матрица распределения. Все распределения, в которых предложение или спрос равны 0, должны быть равны нулю |  | Проверяет обработку нулевых значений. |
| **TC\_09** | Отсутствует Входной сигнал Питания | ,200,150 | 180,120,150 | 5,8,4\n3,7,9\n6,2,5 | Сообщение об ошибке: ValueError: не удалось преобразовать строку в float: |  | Проверьте недостающие данные. |
| **TC\_10** | Отсутствующий Ввод спроса | 100,200,150 | ,120,150 | 5,8,4\n3,7,9\n6,2,5 | Сообщение об ошибке: ValueError: не удалось преобразовать строку в float: |  | Проверьте недостающие данные. |
| **TC\_11** | Неверный ввод затрат (пропущенные цифры) | 100,200,150 | 180,120,150 | 5,8,4\n3,7, \n6,2,5 | Сообщение об ошибке: ValueError: не удалось преобразовать строку в float: |  | Проверьте недостающие данные. |
| **TC\_12** | Неверный ввод затрат (текст вместо цифр) | 100,200,150 | 180,120,150 | a,b,c\nd,e,f\ng,h,i | Сообщение об ошибке: ValueError: не удалось преобразовать строку в float: |  | Проверьте неправильный тип данных. |
| **TC\_13** | Пустой вход питания | “ | 180,120,150 | 5,8,4\n3,7,9\n6,2,5 | Сообщение об ошибке: ValueError: недостаточно значений для распаковки (ожидалось 1, получено 0) |  | Проверьте Пустой запас. |
| **TC\_14** | Пустой Ввод запроса | 100,200,150 | “ | 5,8,4\n3,7,9\n6,2,5 | Сообщение об ошибке: ValueError: недостаточно значений для распаковки (ожидалось 1, получено 0) |  | Проверьте Пустой запрос. |
| **TC\_15** | Отсутствие затрат на ввод | 100,200,150 | 180,120,150 |  | Сообщение об ошибке: ValueError: недопустимый литерал для float(): |  | Проверьте Пустую матрицу затрат. |
| **TC\_16** | Различная длина Поставки и стоимость | 100,200 | 180,120,150 | 5,8,4\n3,7,9\n6,2,5 | Numpy выдаст исключение, потому что supply и costs будут иметь разные размеры. |  | Проверьте несоответствие размеров. |
| **TC\_17** | Различные требования к длине и стоимости | 100,200,150 | 180,120 | 5,8,4\n3,7,9\n6,2,5 | Numpy выдаст исключение, потому что demand и costs будут иметь разные размеры. |  | Проверьте несоответствие размеров. |

Задание №4

import tkinter as tk  
import tkinter.messagebox as messagebox  
import numpy as np  
  
  
def min\_element\_method(supply, demand, costs):  
  
 supply = np.array(supply, dtype=float)  
 demand = np.array(demand, dtype=float)  
 costs = np.array(costs, dtype=float)  
  
 allocation\_matrix = np.zeros\_like(costs, dtype=int)  
 total\_cost = 0  
  
 num\_suppliers = len(supply)  
 num\_consumers = len(demand)  
  
 while np.any(supply > 1e-6) and np.any(demand > 1e-6):  
 valid\_indices = np.where((supply > 1e-6)[:, None] & (demand > 1e-6))  
  
 if not valid\_indices[0].size:  
 break  
  
 masked\_costs = costs[valid\_indices]  
 min\_cost\_index = np.argmin(masked\_costs)  
 supplier\_index = valid\_indices[0][min\_cost\_index]  
 consumer\_index = valid\_indices[1][min\_cost\_index]  
  
 allocated\_quantity = min(supply[supplier\_index], demand[consumer\_index])  
  
 allocation\_matrix[supplier\_index, consumer\_index] = int(allocated\_quantity)  
 supply[supplier\_index] -= allocated\_quantity  
 demand[consumer\_index] -= allocated\_quantity  
 total\_cost += costs[supplier\_index, consumer\_index] \* allocated\_quantity  
  
 if supply[supplier\_index] <= 1e-6:  
 costs[supplier\_index, :] = np.inf  
 if demand[consumer\_index] <= 1e-6:  
 costs[:, consumer\_index] = np.inf  
  
 return allocation\_matrix, total\_cost  
  
  
def calculate\_plan():  
 try:  
 supply = [float(x) for x in entry\_supply.get().split(',')]  
 demand = [float(x) for x in entry\_demand.get().split(',')]  
 costs\_str = text\_area.get("1.0", "end-1c")  
 costs = [[float(x) for x in row.split(',')] for row in costs\_str.splitlines()]  
 allocation\_matrix, total\_cost = min\_element\_method(supply, demand, costs)  
  
 output\_text = "Опорный план:\n"  
 for row in allocation\_matrix:  
 output\_text += str(row) + '\n'  
 output\_text += f"\nОбщая стоимость: {total\_cost}"  
 output\_label.config(text=output\_text)  
  
 except ValueError as e:  
 output\_label.config(text=f"Ошибка: {e}")  
  
  
root = tk.Tk()  
root.title("Построение опорного плана транспортной задачи")  
  
# Ввод данных  
label\_supply = tk.Label(root, text="Запасы поставщиков (через запятую):")  
entry\_supply = tk.Entry(root)  
  
label\_demand = tk.Label(root, text="Потребности потребителей (через запятую):")  
entry\_demand = tk.Entry(root)  
  
label\_costs = tk.Label(root, text="Матрица стоимостей (через запятую, строки с новой строки):")  
text\_area = tk.Text(root, height=10, width=30)  
  
calculate\_button = tk.Button(root, text="Рассчитать", command=calculate\_plan)  
  
output\_label = tk.Label(root, text="")  
  
label\_supply.pack()  
entry\_supply.pack()  
label\_demand.pack()  
entry\_demand.pack()  
label\_costs.pack()  
text\_area.pack()  
calculate\_button.pack()  
output\_label.pack()  
  
root.mainloop()