# Блочные тесты

## 

## Тест №1. Позитивный

* Описание: проверка правильной работы функции getCoefficient
* Функция: getCoefficient(int[,] prices, int r, int c) – поиск коэффициента нуля
  + Входные данные:
    - int[,] prices – стоимость путей {int.MaxValue, 0, 0}{0, int.MaxValue, 0} {0, 4, int.MaxValue}
    - int r – номер строки: 0
    - int c – номер столбца: 1
* Ожидаемый результат: Функция вернет значение 4
* Статус: Пройдено

## Тест №2. Негативный

* Описание: проверка правильной работы функции getCoefficient
* Функция: getCoefficient(int[,] prices, int r, int c) – поиск коэффициента нуля
  + Входные данные:
    - int[,] prices – стоимость путей {int.MaxValue, 0, 1}{0, int.MaxValue, 2} {3, 0, int.MaxValue}
    - int r – номер строки: 1
    - int c – номер столбца: 0
* Ожидаемый результат: Функция вернет значение не 4
* Статус: Пройдено

## Тест №3. Позитивный

* Описание: проверка правильной работы функции GetCost
* Функция: GetCost(Dictionary<int, int> result , int[,] testSources) – поиск стоимости пути
  + Входные данные:
    - Dictionary<int, int> result – пары координат на матрице путей:   
      {0, 1}{1, 2} {2, 0}
    - int[,] testSources – матрица стоимостей для теста (принимает null если вызвана не из теста): {int.MaxValue, 3, 4}{2, int.MaxValue, 3}   
      {1, 5, int.MaxValue}
* Ожидаемый результат: Функция вернет значение 7
* Статус: Пройдено

## Тест №4. Негативный

* Описание: проверка правильной работы функции GetCost
* Функция: GetCost(Dictionary<int, int> result , int[,] testSources) – поиск стоимости пути
  + Входные данные:
    - Dictionary<int, int> result – пары координат на матрице путей:   
      {0, 1}{2, 0} {1, 2}
    - int[,] testSources – матрица стоимостей для теста (принимает null если вызвана не из теста): {int.MaxValue, 3, 4}{2, int.MaxValue, 3}   
      {1, 5, int.MaxValue}
* Ожидаемый результат: Функция вернет значение не -1
* Статус: Пройдено

# Интеграционные тесты

## Тест №1. Позитивный

* Взаимодействие подсистем: Подсистема поиска координат подходящих нулей findBestZeros(int[,] prices, List<int> TestList) использует значения, найденные с помощью функции getCoefficient(int[,] prices, int r, int c)
* Начальное состояние: введен массив стоимостей пути и по нему рассчитан лист коэффициентов
* Входные данные:
  + int[,] prices – массив стоимости путей
  + List<int> TestList – лист коэффициентов для каждого нуля (создается при помощи функции getCoefficient). Принимает null если вызван не из теста
* Ожидаемый результат: Функция вернет лист со значениями {1, 6}
* Статус: Пройдено

## Тест №2. Негативный

* Взаимодействие подсистем: Подсистема поиска координат подходящих нулей findBestZeros(int[,] prices, List<int> TestList) использует значения, найденные с помощью функции getCoefficient(int[,] prices, int r, int c)
* Начальное состояние: введен массив стоимостей пути и по нему рассчитан лист коэффициентов
* Входные данные:
  + int[,] prices – массив стоимости путей
  + List<int> TestList – лист коэффициентов для каждого нуля (создается при помощи функции getCoefficient). Принимает null если вызван не из теста
* Ожидаемый результат: Функция вернет лист со значением не {6}
* Статус: Пройдено

# Аттестационные тесты

## Тест №1. Позитивный

* Начальное состояние: пользователь вводит данные в файле input.txt:
  + Размер матрицы
  + Значения матрицы (там, где бесконечность ставить -1)
* Действие: Рассчитать по этой матрице наикратчайший путь и его стоимость
* Ожидаемый результат: В файле output.txt значение наикратчайшего пути и сам путь
* Статус: Пройдено