Домашнее задание № 1а
Алгоритмы и структуры данных

Проход Грэхема

Авторы задания: команда курса «Алгоритмы и структуры данных 2020».
Вопросы по заданию направлять:
Рахмановский Андрей Дмитриевич adrakhmanovskiy_1@edu.hse.ru
или в канал в MS Teams “**Домашнее задание 1а**”

Это домашнее задание выполняется на оценку.

Отправка решения. Результатом решения должен быть один файл на языке C# (.cs), а также файлы тестов (входные и выходные данные). Именно файл на языке C# (.cs) и тесты будут оцениваться.
Решение нужно загрузить в LMS в **Проект 1а** по дисциплине
Алгоритмы и структуры данных 2020 уч. год Б 2 курс (код 131940, ОП: М090304ПИНЖ)
При выявлении плагиата будет выставлена оценка 0 баллов

Дедлайн. 30 сентября 2020 г. 23:59 МСК
Пожалуйста, загружайте решения заранее, чтобы не возникало технический препятствий в последний момент.

Что нужно сделать?

1. Решите задание (один файл C#).
2. Напишите 5 своих тестов (пары файлов входных и выходных данных) к задаче, пронумеруйте их соответственно. Положите файлы с входными данными в папку input, а с выходными в папку output.
3. Создайте папку с решением по шаблону <Имя>_<группа>, например RahmanovskiyAndrey_161. Положите в неё папки с тестами и один файл .cs с решением задачи.
4. Упакуйте всё в Zip-папку и отправьте в LMS (размер файла должен быть не более 1МБ).
5. Готово!

Внимание: не публикуйте свое решение в открытом доступе, напр. на GitHub либо других открытых источниках. Это действие будет приравниваться к плагиату и повлечет за собой оценку 0 баллов за домашнее задание (в том числе при обнаружении такового факта уже после выставления положительной оценки).

Условие2

Однажды жил-был некоторый человек по имени Поликарп. Он был самым обыкновенным человеком, жил в городе, учился и работал. Однако, в какой-то момент он решил стать отшельником. Пошел он в лес, искать себе новое место для жилья. Ходил он по разным лесам и полям, пока не вышел на чудесную опушку, на которой было **N** деревьев. Это место ему очень понравилось, и он решил здесь обосноваться. Первым делом он захотел обозначить свою новую территорию соединив веревкой некоторые деревья так, чтобы из них получился выпуклый многоугольник, а все остальные деревья на этой опушке должны были оказаться внутри данного многоугольника. При этом веревка у него не бесконечная, поэтому получившийся выпуклый многоугольник должен быть наименьшим. Опушка представляет из себя плоскость, а каждое дерево задаётся двумя целочисленными координатами – **x** и **y**. Однако Поликарп не проходил курс Алгоритмы и структуры данных и не может понять, между какими деревьями надо проводить веревку, а между какими – нет, поэтому он обратился к вам за помощью!

Формат входных данных

На первой строке файла находится натуральное число N (3<= N <= 10^3) - количество деревьев.
Далее в N строках вводятся по 2 целых неотрицательных числа – координаты деревьев на этой опушке x и y (x,y <= 10^4).

Формат выходных данных

Деревья необходимо выводить в порядке обхода против часовой стрелки либо по часовой стрелке (зависит от параметра командной строки, см. ниже), но всегда начиная с дерева с минимальной y-координатой (в случае, если таких деревьев несколько, необходимо выбрать дерево с минимальной x-координатой) – справедливо для обоих форматов (см. ниже).

Формат №1 (Plain)

На первой строке выходного файла находится единственное число – сколько деревьев необходимо связать между собой.

Далее на каждой строке выведите координаты каждого дерева x и y, которое будет вершиной выпуклого многоугольника.

Формат №2 (WKT, Well-Known Text)

На первой строке выходного файла находится одна строка, которая повторяет входные данные в формате MULTIPOINT:
MULTIPOINT <пробел> ((x1 <пробел> y1), <пробел> (x2 <пробел> y2), …)

На второй строке выходного файла находится одна строка в формате POLYGON, которая содержит результат – минимальный выпуклый многоугольник:
POLYGON <пробел> ((x1 <пробел> y1, x2 <пробел> y2, …, x1 <пробел> y1))
Обратите внимание на повтор в конце первой координаты.

Пример (обход против часовой стрелки, формат plain)

Входные данные	Выходные данные
4 0 0 3 4 3 1 6 0	3 0 0 6 0 3 4
5 0 0 0 1 0 2 1 0 2 0	3 0 0 2 0 0 2
8 0 0 0 10 10 10 1 9 1 1 2 9 10 9 10 0	4 0 0 10 0 10 10 0 10

Пример (обход по часовой стрелке, формат wkt)

Входные данные	Выходные данные
4 0 0 3 4 3 1 6 0	MULTIPOINT ((0 0), (3 4), (3 1), (6 0)) POLYGON ((0 0, 3 4, 6 0, 0 0))
5 0 0 0 1 0 2 1 0 2 0	MULTIPOINT ((0 0), (0 1), (0 2), (1 0), (2 0)) POLYGON ((0 0, 0 2, 2 0, 0 0))

Примечание
В данном задании необходимо реализовать следующее:

- Класс точки, который будет содержать:
  - Приватные поля для хранения координат
  - Конструктор для задания координат
  - Геттеры для получения доступа к координатам
- Класс стека, который будет содержать:
  - Приватное поле для хранения данных
  - Публичное константное поле максимального допустимого размера стека
  - Конструктор для задания максимального размера стека (может быть меньше либо равен максимальному допустимому размеру стека)
  - Методы push, pop, top, nextToTop, size, isEmpty
- Также необходимо:
  - Генерировать исключение при попытке добавить элемент в стек, когда достигнуто максимально возможное количество элементов
  - Генерировать исключение при попытке удалить элемент из пустого стека

Как будут оцениваться работы? Решения будут проверяться на наборах тестов, каждый тест это два файла - один для входных данных, другой для выходных. Таким образом, ввод и вывод данных происходит через файлы.

Будут оценены: корректность кода, составленные тесты (граничные условия, сложные случаи и т.п.), декомпозиция, аккуратность кода. Возможны устные собеседования при сомнении в самостоятельности выполнения работы.

В приложенной к условию задачи папке расположено 5 тестов (формат plain, обход против часовой стрелки), на которых можно проверить работоспособность решения, однако стоит учитывать, что прохождение приложенных тестов не гарантирует полное отсутствие ошибок в решении (решения будут также проверяться и на других, приватных тестах). Файлы входных данных именуются по принципу “test<номер теста>.txt”, файлы выходных данных (или ответов) называются “answer<номер теста>.txt”.

Названия файлов для входных данных и выходных данных должны быть указаны в командной строке. Кроме того, в параметрах командной строки могут указываться не только имена файлов, но и пути к ним. Например, так будет выглядеть тестирование программы RahmanovskiyAndrey_161.cs:

csc RahmanovskiyAndrey_161*.cs*

RahmanovskiyAndrey_161 cw plain *test1.txt answer1.txt*

либо

RahmanovskiyAndrey_161 cc wkt *path/to/test1.txt path2/to/answer1.txt*

где

1ый параметр – порядок обхода деревьев,

cw – clockwise (по часовой стрелке),

cc – counterclockwise (против часовой стрелки)

2ой параметр – формат выходных данных

plain – Формат №1

wkt – Формат №2 (Well-Known Text)

3ий параметр – путь к входному файлу

4ый параметр – путь к выходному файлу

Пример использования аргументов командной строки в C#:
https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/main-and-command-args/command-line-arguments