Общие условия

- В тестовом задании три задачи, с возрастанием уровня сложности от первой к последней.

- При соблюдении срока сдачи и требований по оформлению результатов решения задач приниматься и проверяться будут решения как минимум двух задач. Но если удастся решить все, это будет плюсом в общей оценке.

- Если не удалось закончить решение одной задачи, можно прислать незаконченное или неработающее решение, при условии, что полностью выполнены две другие.

Общее требование к оформлению результата

- Разрешено использовать следующие языки программирования:
  JAVA, Kotlin, Swift и C#.

- Решение каждой задачи должно быть в отдельном TXT-файле (для третьей задачи - 2 файла). Имя файла должно соответствовать шаблону:

  Фамилия.Имя.Язык программирования.Задача<номер задачи>.txt

  Пример: *Иванов.Иван.Kotlin.Задача1.txt*

  Для третьей задачи в конец имени файла с результатом работы алгоритма добавить слово “Результат”.

  Пример: *Иванов.Иван.Kotlin.Задача3.Результат.txt*

  Важно: Указывайте те же имя и фамилию, с которыми регистрировались на тренинг, и присылайте решенные тестовые с почты, которую указали при регистрации.

Пожелания к качеству кода

- Наименования классов, функций, методов, переменных и др. должны быть лаконичными, но и в то же время нести смысловую нагрузку. Сокращения не приветствуются.

- Хорошо отформатированный код без лишних комментариев будет плюсом, и наоборот – неаккуратный и плохо читаемый код будет минусом при оценке решения задачи.

Задача 1. Проверка лотерейного билета

Написать функцию, которая на вход принимает номер лотерейного билета и возвращает булевское значение, выигрышный билет или нет.

Входные параметры:

- Функция должна принимать в качестве входного параметра номер (число) лотерейного билета, количество цифр которого чётное от 2 до 8.

Условия:

- Выигрышным считается билет, сумма цифр левой половины номера которого равна сумме цифр правой половины.

Возвращаемый результат:

- только булевское значение: true – в случае, если билет выиграл; false – если не выиграл.

Оформление результата выполнения задачи:

- Листинг функции в одном TXT-файле.

Задача 2. Разработать цифровой аналог автопарка

Условия:

- В автопарке должны быть представлены различные виды транспорта (грузовой, пассажирский, грузопассажирский).

- У каждого транспортного средства должны быть свои характеристики, как общие (например, год выпуска, марка и модель, вид используемого топлива, его расход ...), так и специфичные (например, объём кузова и грузоподъёмность для транспорта, перевозящего грузы, и пассажировместимость – для перевозящего пассажиров).

- Аналогично с методами: должны быть как общие (например, заправить и отремонтировать), так и специфичные (например, продезинфицировать салон для автобусов и опломбировать кузов для грузовиков).

- Грузовые автомобили должны иметь разные типы кузовов и, соответственно, разные варианты перевозимых грузов (например, в тентованных можно перевозить промтовары, в рефрижераторах – промтовары и скоропортящиеся продукты, в цистернах – жидкости и т.д.).

- Также следует предусмотреть выполнение заказов на перевозки. Заказ должен включать начальный и конечный пункты, а также объём, вес и тип груза для грузоперевозок или количество пассажиров для пассажироперевозок. Должна быть возможность погрузить и разгрузить заказ в/из конкретной машины. У каждого транспортного средства должна быть возможность просмотреть свободную грузоподъёмность, объём (либо количество мест) и заказы, которые на данный момент загружены в машину.

Результат разработки:

- Объектно-ориентированное описание автопарка.

- Требуется промоделировать работу (создать несколько транспортных средств, заправить и обслужить их, загрузить/разгрузить несколько заказов). В идеале – с использованием автоматизированного цикла.

Оформление результата выполнения задачи:

- Листинг описания и примера с моделированием работы в одном TXT-файле.

Задача 3. Проверка и оптимизация резов стекла

Имеется станок ЧПУ для раскроя листов стекла. На него подаются прямоугольные листы стекла, и режущий инструмент станка (далее – резец), осуществляет процесс резки. Резец станка имеет 2 режима перемещения: “холостой ход”, когда он перемещается в поднятом состоянии; и “рабочий ход”, когда он перемещается и непосредственно режет стекло.

Программирование работы станка осуществляется заданием списка отрезков, по которым происходит перемещение резца. Список отрезков включает в себя непосредственно входной набор отрезков и некоторые стороны фигур из входного набора фигур.

Требуется реализовать алгоритм с функциями:

- преобразования сторон фигур в отрезки с координатами в СКЛ;

- поиска и исключения из списка отрезков, которые накладываются на другие отрезки;

- оптимизации списка отрезков для уменьшения расстояния “холостого хода” резца.

Алгоритм должен принимать в качестве входных параметров:

- ***Список фигур*** *– приведен в разделе “Входные данные для задачи 3”.*

  Фигура описывается 4-мя точками на плоскости с координатами (X, Y), образующими четырёхугольник. Координаты точек фигуры даны в системе координат фигуры (СКФ).
  Положение фигуры на листе задаётся координатами начала системы координат фигуры (СКФ) в системе координат листа (СКЛ). Начало системы координат листа (СКЛ) совпадает с нижним левым углом листа.

- ***Список отрезков на плоскости листа*** *приведен в разделе “Входные данные для задачи 3”.*

  Отрезки заданы двумя точками, точка определяется координатами X, Y (СКЛ).

Примечание: все координаты целочисленные, все отрезки являются прямолинейными.

Условия:

Список отрезков передается на станок резки стекла. Для успешного и экономного выполнения операции резки стекла нужно соблюсти следующие условия:

- стороны фигур должны быть преобразованы в отрезки и добавлены в общий список отрезков;

- станок не должен дважды резать по одной линии. То есть исключить из списка отрезки, которые полностью накладываются на другие отрезки;

- необходимо сократить расстояние “холостого хода” резца. То есть оставшиеся отрезки должны быть расположены в списке таким образом, чтобы начало следующего отрезка являлось ближайшей точкой к концу текущего отрезка. Для этого можно менять положение отрезков в списке и менять местами начальную и конечную точки подходящего отрезка.

Пояснение

Резец начинает движение из начала координат СКЛ (точка (0,0) совпадает с левым нижним углом листа) и двигается по прямой к начальной точке первого отрезка из списка в поднятом состоянии (“холостой ход”). В этой точке резец опускается и, двигаясь к конечной точке отрезка, прорезает стекло. Далее резец поднимается и перемещается по прямой к начальной точке следующего отрезка и т. д.

Результат работы алгоритма:

- Список отрезков, удовлетворяющий условиям и заданный координатами X1, Y1, X2, Y2 в СКЛ.

Оформление результата выполнения задачи:

- Листинг алгоритма в одном TXT-файле.
- TXT-файл с результатом работы алгоритма: только список отрезков, удовлетворяющий условиям. Каждая строка в файле – отдельный отрезок, заданный координатами X1, Y1, X2, Y2 в СКЛ.

Входные данные для задачи 3

Список отрезков:
Каждая строка отдельный отрезок, заданный координатами X1, Y1, X2, Y2 в СКЛ:

```
15, 0, 15, 3210
0, 15, 6000, 15
1500, 0, 1500, 3210
15, 1015, 1500, 1015
15, 2015, 1500, 2015
15, 3015, 1500, 3015
2550, 0, 2550, 3210
1500, 1415, 2550, 1415
1500, 2815, 2550, 2815
3991, 0, 3991, 3210
2550, 515, 3991, 515
2550, 1015, 3991, 1015
2550, 1515, 3991, 1515
2550, 2015, 3991, 2015
2550, 2765, 3991, 2765
3250, 2015, 3250, 2765
4789, 0, 4789, 3210
3991, 1515, 4789, 1515
3991, 3015, 4789, 3015
5843, 0, 5843, 3210
4789, 1123, 5843, 1123
5316, 15, 5316, 1123
```

Список фигур. Каждая фигура описана 4-мя точками (X, Y) в СКФ

```
Фигура 1:
4 точки фигуры:
0, 0
1470, 0
1200, 1000
0, 1000
Положение фигуры в СКЛ:
15, 15

Фигура 2:
4 точки фигуры:
0, 0
1470, 0
1200, 1000
0, 1000
Положение фигуры в СКЛ:
15, 1015
```

Фигура 3:
4 точки фигуры:
15, 0
1485, 0
1485, 1000
285, 1000
Положение фигуры в СКЛ:
15, 2015

Фигура 4:
4 точки фигуры:
0, 0
798, 0
798, 1485
0, 1000
Положение фигуры в СКЛ:
3991, 15

Фигура 5:
4 точки фигуры:
0, 0
798, 0
798, 1200
0, 1485
Положение фигуры в СКЛ:
3991, 1515

Фигура 6:
4 точки фигуры:
15, 0
685, 0
600, 735
150, 735
Положение фигуры в СКЛ:
2550, 2015

Пример к задаче 3.

Входные данные:

Список отрезков:

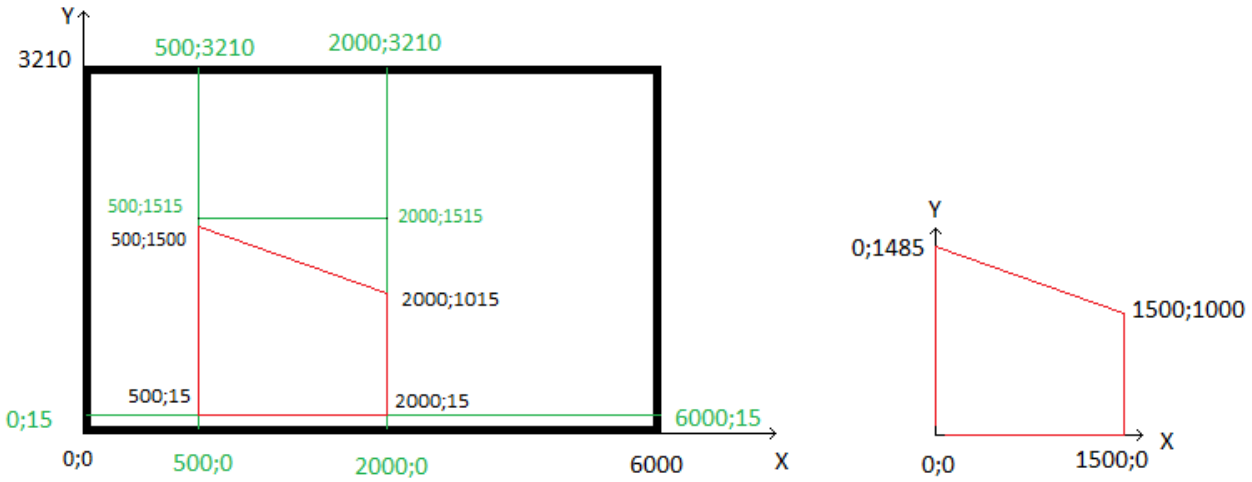
```
500, 0, 500, 3210
0, 15, 6000, 15
2000, 0, 2000, 3210
500, 1515, 2000, 1515
```

Фигура:

```
4 точки фигуры:
0, 0
1500, 0
1500, 1000
0, 1485
Положение фигуры в СКЛ:
500, 15
```

Схематическое изображение чертежа листа, отрезков и фигуры для данного примера:

Размер листа: ширина = 6000 мм, высота = 3210 мм.



Результат работы алгоритма и пример того, что должно быть во втором файле:

```
0, 15, 6000, 15
2000, 0, 2000, 3210
500, 3210, 500, 0
500, 1500, 2000, 1015
2000, 1515, 500, 1515
```