1. **Имеется три процедуры:**

Circle(X, Y, L) // рисует круг

Square(X, Y, L) // рисует квадрат

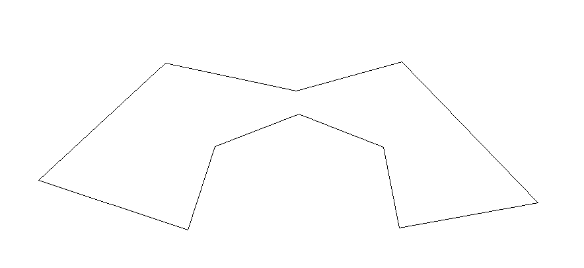
CircleOrSquare(X, Y, L) // рисует круг или квадрат

**Программист перепутал названия у всех процедур так, что ни одна из них не работает по своему названию. Исходный код процедур недоступен. Сколько потребуется запусков и каких процедур, чтобы понять, какая процедура что делает?**

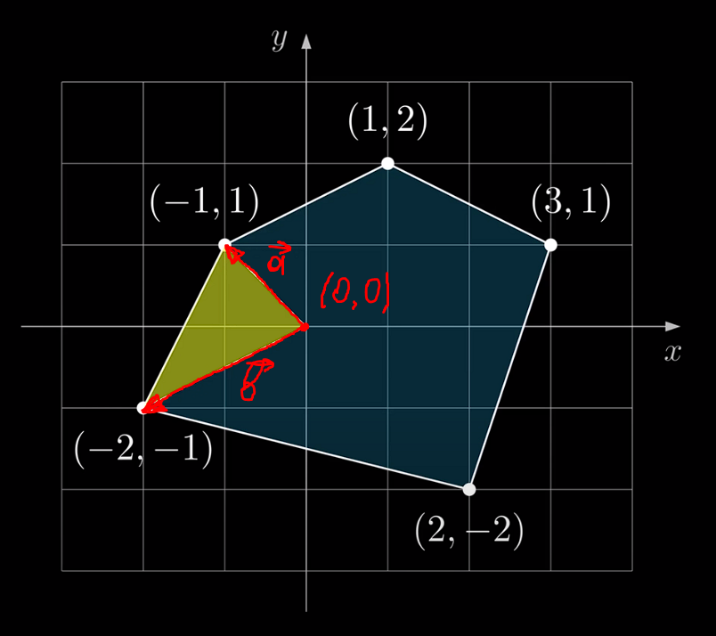
*(1 шаг) Нужно активировать каждую из функций по очереди. Так мы сможем точно идентифицировать одну функцию и убрать ее из рассмотрения (т.е., например после активации получили картину: CircleOrSquare – нарисовал квадрат, Circle – нарисовал круг, Square – нарисовал круг. Отсюда ясно что CircleOrSquare это первоначально была функция Square).*

*(2 шаг) Далее, мы запускаем две оставшиеся функции, пока не увидим различающиеся ответы (Возвращаясь к примеру: Circle – нарисовал круг, Square – нарисовал круг. Запускаем их пока одна из них не нарисует квадрат – это и будет перепутанная функция CircleOrSquare).*

*Таким Образом, понадобится 5 + 2k повторений, где k – это количество раз которое будет выполнен 2 шаг*



1. **Предложить алгоритмическое решение задачи нахождения площади произвольного многоугольника, у которого известны координаты узловых точек.**

*Для нахождения площади произвольного многоугольника, я бы воспользовался подходом разбиения многоугольника на треугольники, с последующим нахождением их площадей (триангуляция).*  
  
Можно также использовать подход через использование шнуровки Гаусса, заключающегося в нахождении скалярного произведения векторов.

1. **Каким будет результат работы программы? Напишите, что она делает**.

program Transpose;

type TMatr = array[0..3, 0..3] of Integer;

var matr: TMatr = (

(1, 2, 3, 4),

(1, 2, 3, 4),

(1, 2, 3, 4),

(1, 2, 3, 4)

);

i, j, k: Integer;

begin

for i:=0 to 3 do

for j:=0 to 3 do

begin

k:=matr[i, j];

matr[i,j]:=matr[j,i];

matr[j,i]:=k;

end;

for i:=0 to 3 do

begin

for j:=0 to 3 do

write(matr[i, j]:4);

writeln;

end;

end.

*Данный код транспонирует матрицу. Элементы симметричные относительно главной диагонали меняются местами. Вывод будет примерно такой  
1 1 1 1*

*2 2 2 2*

*3 3 3 3*

*4 4 4 4*

1. **В городе работает телефонное справочное бюро, в котором имеется список телефонов 500000 горожан. Каждый день горожане звонят и просят по номеру телефона определить владельца. Какие меры должен предпринять специалист IT-отдела бюро, чтобы ответ оператора на запрос выполнялся максимально быстро? Рассмотреть варианты с базами данных и без.**

*Без баз данных:  
Можно воспользоваться ассоциативными структурами данных, например хештаблица. Нам нужно будет использовать номер телефона как ключ (или аргумент для хэш-функции), а Имя как значение хранящееся по этому ключу. Это позволит получать данные о владельце за O(1)*

*Также, можно хранить номера ввиде дерева (корень дерева это условно +7, а дальше идет разветвление по поддеревьям. Если брать наши номера (7 999 888 77 66) то дерево получится высотой 11. В узлах дерева будет храниться информация о человеке) что также облегчит поиск. В отличие от хэш таблицы, в дереве не должно случаться ошибочного определения владельца (что происходит из-за хэш-функции), конечно же, если все номера уникальны.  
  
Также, можно отдельно хранить самые частые номера (тоже в хэш-таблице например, или даже в массиве)*

*с базами данных:*

*Для баз данных я вижу только, наверное, возможность индексации таблицы по полю номер телефона.*

*А также, можно, использовать параллельные запросы, чтобы к примеру обрабатывать нисколько клиентов одновременно*