MAC-адреса кодируют номер предприятия, а остальные — серийный номер устройства. Узнать MAC-адрес на Windows-ЭВМ можно командой *getmac*, набранной в командной строке. При этом на экран шестнадцатиричные пары будут выданы разделенными символом тире, например, A4-CE-D8-F4-AB-FB, (в Linux для этого используется символом двоеточие). Реализуйте функцию, получающую MAC-адрес в формате текстовой строки и значение n и возвращающую номер предприятия и серийный номер устройства в виде двух десятичных чисел.

1.206. Для идентификации компьютеров в разных подсетях используется четырехбайтовый IP-адрес, для удобства записываемый в виде четырех десятичных чисел разделенных точкой, например, 192.168.0.1. Таблицу соответствий между MAC-адресами и IP-адресами можно получить, набрав в командной строке команду *arp -a*. Первые n бит IP-адреса кодируют адрес сети, а остальные — адрес узла в сети. Реализуйте функцию, получающую IP-адрес в виде текстовой строки и величину n и и возвращающую два четырехбайтовых массива с адресом сети и адресом узла соответственно. Распечатайте найденные адрес сети и адрес узла в виде четырех десятичных чисел разделенных точкой. Модифицируйте функцию так, чтобы она допускала ввод IP-адреса и величины n форме одной строки, содержащей т.н. краткую слешнотацию: 192.168.161.1/24.

1.207. В терминологии сетей TCP/IP маской сети называют четырехбайтовый адрес, у которого первые n бит единицы, а остальные — нули. Побитно "перемножив" IP-адрес и маску — получим адрес сети, побитно "перемножив" IP-адрес и "инвертированную" маску — найдем адрес узла в сети.

Реализуйте функцию, получающую IP-адрес и маску сети в виде двух четырехбайтовых массивов, и возвращающую четырехбайтовые массивы с адресом сети и адресом узла. Отметим, что для адресации компьютеров в подсети не используются два адреса: адрес сети, т.е. адрес в котором все биты, отсекаемые маской, равны 0, и т.н. широковещательный адрес, в котором все биты, отсекаемые маской, равны 1. Включите в функцию проверку на корректность полученного адреса узла.

1.8 Задачи на обработку множества точек

В следующих задачах предполагается, что в файле записано несколько пар чисел, которые можно рассматривать как координаты множества точек на плоскости или как координаты множества концов отрезков на прямой. Требуется составить программы, которые читают исходные данные и отвечают на поставленные вопросы, или выдают значения параметров требуемых геометрических объектов. Следует стремиться к построению минимальных по сложности алгоритмов (сложность оценивается порядком числа операций в зависимости от числа точек или отрезков).

1.208. Множество точек определяет ломаную. Имеет ли она самопересечения?

1.209. Множество точек определяет многоугольник. Является ли он выпуклым?

1.210. Множество точек определяет многоугольник. Определите его диаметр (т.е. наибольшую диагональ). Решение должно иметь сложность $O(N)$, где N — число вершин многоугольника.

1.211. Дано множество отрезков. Покрывает ли их объединение заданный отрезок $[a, b]$?

1.212. Дано множество отрезков. Найдите точку, которая принадлежит наибольшему количеству отрезков, определите это количество.

1.213. Дано множество точек. Найдите центр и радиус минимального круга, который содержит все эти точки.

1.214. Дано множество отрезков. Выберите из него и распечатайте связное подмножество отрезков, объединение которых дает отрезок наибольшей длины.

1.215. Дано множество точек. Из этих точек как из центров начинают одновременно и с одинаковой скоростью "расти" круги (т.е. точки являются центрами, а радиусы увеличиваются с одинаковой скоростью). Если два круга сталкиваются, то их рост прекращается. Определить радиусы всех получившихся кругов после того как их рост прекратится.

1.216. Окружность разделена на m равных дуг и внутри каждой дуги на окружности отмечено некоторое количество точек (возможно, нулевое). Требуется передвинуть эти точки на окружности так, чтобы были выполнены следующие условия:

1. каждая точка должна по возможности наименее удаляться от своей исходной позиции;

2. ни одна точка не должна покидать свою первоначальную дугу;

3. точки должны сохранять исходный порядок, а угловое расстояние между любыми двумя соседними точками не должно быть меньше заданной величины δ (для разрешимости данной задачи можно считать, что $\delta < 2\pi/mn$, где n — наибольшее количество точек, приходящихся на одну дугу).

1.217. Обобщением задачи 1.216 является задача о размещении названий на географической карте. Пусть заданы координаты нескольких точек на плоскости (населенные пункты) и с каждой точкой связан некоторый прямоугольник, задаваемый своей длиной и шириной (он ограничивает текст наименования населенного пункта). Требуется разместить каждое наименование рядом с соответствующей ему точкой так, чтобы наименования (прямоугольники) не накладывались друг на друга.

1.218. Множество точек определяет многоугольник (возможно невыпуклый, но без самопересечений). Определите находится ли заданная точка внутри или вне этого многоугольника.

1.219. Пусть плоские выпуклые многоугольники представляются следующим типом данных:

```
struct Polygon {int n; double *x,*y;};
```

где n — число вершин многоугольника, x, y — указатели на массивы координат вершин. Реализуйте функции, обеспечивающие работу с многоугольниками, как с объектами:

```
double  Perimeter(struct Polygon p);                          /* периметр */
double  Area(struct Polygon p);                               /* площадь */
struct Polygon * Clip(struct Polygon a, struct Polygon b);/* пересечение */
int     Equal(struct Polygon a, struct Polygon b);            /* равны ? */
int     Convex(struct Polygon a);                             /* выпуклый ? */
```

Считаем, что пересечение непересекающихся многоугольников пусто. Функция `Clip` создает новый многоугольник, т.е. выделяет для него память и заполняет требуемыми значениями.

1.220. Пусть в трехмерном пространстве определена плоскость — задан вектор нормали и трехмерные координаты одной точки. Реализуйте функцию, которая вычисляет координаты ортогональной проекции точки пространства на эту плоскость.

1.221. Пусть в трехмерном пространстве дана плоскость — два базисных вектора и трехмерные координаты одной точки. Реализуйте функцию, которая вычисляет координаты ортогональной проекции точки пространства на эту плоскость.

1.222. Пусть в трехмерном пространстве дана плоскость с локальной системой координат — два базисных вектора и трехмерные координаты точки на плоскости, являющейся началом координат. Реализуйте функцию, которая вычисляет локальные координаты ортогональной проекции точки пространства на эту плоскость.

1.9 Рекурсия.

1.223. Напишите рекурсивную и последовательную реализации вычисления функции $n!$, сравните их эффективность (см. 1.16, 1.30).

1.224. Напишите рекурсивную реализацию для вычисления дисперсии $D = \frac{1}{n}\sum_{i=1}^{n}(x_i - M)^2$ элементов числовой последовательности. Сравните эффективность с 1.45).

1.225. Оцените размер программного стека, выделяемого в Вашей системе.
Решение. В программном стеке для каждой вызываемой функции хранятся все локальные переменные класса `auto`, возвращаемое значение и адрес оператора возврата. Поэтому каждый рекурсивный вызов функции

```
void  f(int n){
  char buf[1016];
  printf("n=%d\n",n);
  f(n+1);
  return ;
```