**1. Простое задание (по вариантам).**

Нарисуйте дерево и выпишите последовательность бит, получаемую при компрессии алгоритмом Хаффмана слова, заданного номером варианта. Нужно выписать только сжатую последовательность бит, без служебной информации.

| Номер варианта | Слово для компрессии |
|---|---|
| 1, 12, 23 | ПЕРЕДЕЛАТЬ |
| 2, 13, 24 | ПАПОРОТНИК |
| 3, 14, 25 | КОЛОВОРОТ |
| 4, 15, 26 | ГОРГОНЗОЛА |
| 5, 16, 27 | ЗАКОПАННЫЕ |
| 6, 17 | РАЗРАБОТКА |
| 7, 18 | НАЗНАЧАТЬ |
| 8, 19 | ФОТОФОНАРЬ |
| 9, 20 | КОЛОРИФЕР |
| 10, 21 | ЗАКОЛОТИТЬ |
| 11, 22 | ОБОРОНЯТЬ |

**2. Сложное задание (одно на всех, требует кодирования на C++).**

Напишите программу **минимального размера** (по размеру исходного кода), которая выведет в файл указанный ниже код. Требуется, чтобы полностью совпадал не просто текст, но и сохранялось форматирование текста (отступы, регистр и т.д.).

Вывод сделать в текстовый файл out.txt.

Мы будем просто сравнивать два текстовых файла (наш и ваш) и искать отличия.

В приведённом участке кода все отступы – это пробелы (уж не знаю, как PDF сохранит), в концах строк никаких лишних пробелов нет.

**ВНИМАНИЕ!**

1. От вас мы ждём **только один файл .cpp**. Кто пришлёт более одного файла – даже проверять не будем, будем считать, что решение не прислано!
2. Вы выполняете СРС – самостоятельную работу! В том случае, если будут присланы два одинаковых исходника, мы не засчитываем ни первый, ни второй. Поверьте мне на слово, мы сможем увидеть, что два исходника одинаковые (даже если вы возьмёте чужой исходник и, например, поменяете имена переменных). Или делайте самостоятельно, или не присылайте вовсе.
3. Специально повторю, что вывод ОБЯЗАН быть выполнен в файл с именем out.txt.

Участки текста со словами «это выводить не надо» выводить в файл не нужно, это ограничители, чтобы вам было понятнее, откуда и докуда выводить.

```
--- (начать вывод со следующей строки, это выводить не надо)
class Element {
public:
  Element(type t, char *name);
  virtual double get_current() = 0;
  virtual double get_voltage(int pin) = 0;
  virtual void update() = 0;
};

class Element_Resistor : public Element {
public:
  Element_Resistor(char *name);
  virtual double get_current();
  virtual double get_voltage(int pin);
  virtual void update();
};

class Element_Capacitor : public Element {
public:
  Element_Capacitor(char *name);
  virtual double get_current();
  virtual double get_voltage(int pin);
  virtual void update();
};

class Element_Diode : public Element {
public:
  Element_Diode(char *name);
  virtual double get_current();
  virtual double get_voltage(int pin);
  virtual void update();
};
--- (вывода заканчивается на предыдущей строке, это выводить не надо)
```

Ответ нужно прислать в электронном виде на адрес dima@pkims.ru

В заголовке письма прошу написать: «ТА СРС5», следом укажите вашу фамилию и группу.

В письме укажите ваш вариант и приложите картинку и файл с исходным кодом.

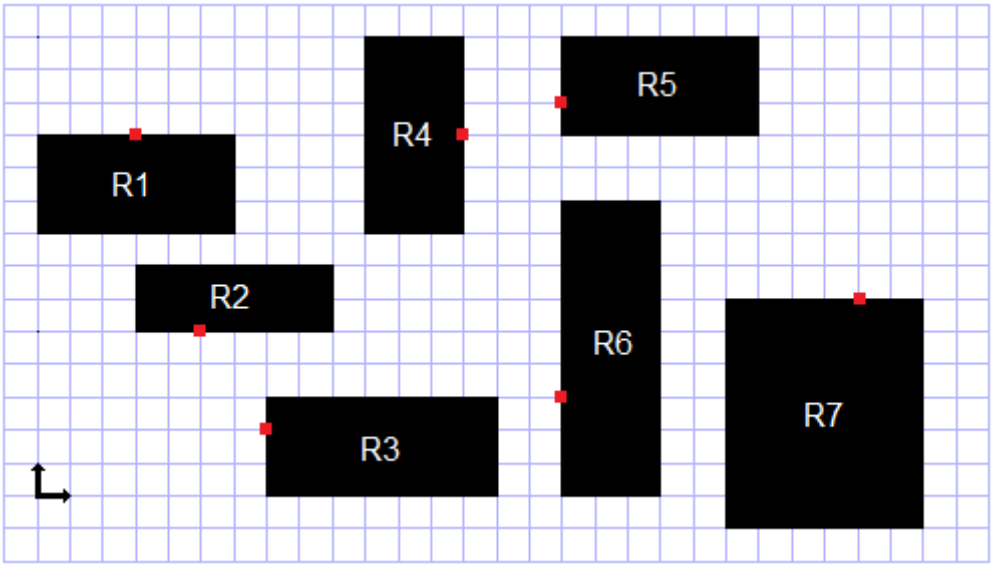
Крайний срок приёма ответов для этого задания – 11 ноября 23:59. Ответы, отправленные позже этого срока, например, 12 ноября в 0:00, проверяться не будут.

*Те, кому нечего делать и кто хочет попрактиковаться в программировании на более серьёзных задачах, крутят ниже.*

По итогам 3 СРС я увидел много примеров присланного кода, хотя этого по заданию и не требовалось. Возможно, вам хочется побольше попрограммировать? Или задания на лабораторную работу слишком простые и их не очень интересно делать? Тогда это задание для вас!

*Важное замечание: выполнение этого задания – на интерес, никаких дополнительных баллов за его выполнение вам дано не будет. Если никто не пришлёт результат – я не расстроюсь.*

Я попытался сымитировать реальную задачу.




На поверхности кристалла кремния расположено некоторое количество элементов, представленных в виде чёрных ящиков. У нас есть информация лишь о том, какие координаты они имеют и о том, какие координаты имеют внешние контакты этих элементов. Эта информация содержится в файле, формат которого представлен в листинге.

```
RECT
NAME R1
PIN 3 3
POINTS 0 11 6 8
END

RECT
NAME R2
PIN 2 0
POINTS 3 7 9 5
END

RECT
NAME R3
PIN 0 2
```

```
POINTS 7 3 14 0
END

RECT
NAME R4
PIN 3 3
POINTS 10 14 13 8
END

RECT
NAME R5
PIN 0 1
POINTS 16 14 22 11
END

RECT
NAME R6
PIN 0 3
POINTS 16 9 19 0
END

RECT
NAME R7
PIN 4 7
POINTS 21 6 27 -1
END
```

Каждый элемент описывается блоком RECT, для которого задаётся имя, координаты пина (контакт блока с внешним миром, координаты пина отсчитываются относительно левого нижнего угла блока) и координаты левого верхнего и правого нижнего угла блока.

Гарантируется, что:

1. все координаты – целочисленные;
2. имена блоков – без пробелов, в одно слово;
3. порядок записей в блоке неизменен;
4. синтаксических ошибок во входном файле нет.

Напишите программу, которая:

1. считает входной файл, передаваемый в качестве аргументов командной строки;
2. определит линейные размеры (ширину и высоту) занимаемой элементами области;
3. определит площадь кристалла, которую займёт эта группа элементов;
4. найдёт минимальное остовное дерево по пинам блоков (красные точки);
5. посчитает суммарную длину межсоединений найденного дерева (используем манхэттенскую метрику).

Постарайтесь применить ООП (структуры/классы) для хранения информации о блоке.