Требования к программам

1. Программа работает с двунаправленным списком объектов типа `student`:

```
class student
  {
  private:
    char * name;
    int    value;
  public:
    ...
  };
class list2_node : public student
  {
  private:
    list2_node * next;
    list2_node * prev;
  public:
    ...
    friend class list2;
  };
class list2
  {
  private:
    list2_node * head;
  public:
    ...
    int read (const char * filename);
    void print (int r);
    int get_length ();
  };
```

   Все функции в задании являются членами класса "список".

2. Программа должна получать все параметры в качестве аргументов командной строки. Аргументы командной строки:

   1) r – количество выводимых значений в списке,
   2) `filename` – имя файла, откуда надо прочитать список.
   3) k – параметр для задачи.

   Например, запуск

```
./a.out 4 a.txt 2
```

   означает, что список прочитать из файла `a.txt`, выводить не более 4-х элементов списка и $k = 2$.

3. Класс "список" должен содержать функцию ввода списка из указанного файла.

4. Ввод списка из файла. В указанном файле находится список в формате:

Слово-1	Целое-число-1
Слово-2	Целое-число-2
…	…
Слово-n	Целое-число-n

где слово – последовательность алфавитно-цифровых символов без пробелов. Концом ввода считается конец файла. Программа должна выводить сообщение об ошибке, если указанный файл не может быть прочитан или содержит данные неверного формата.

5. Решение задачи должно быть оформлено в виде подпрограммы, находящейся в отдельном файле. Получать в этой подпрограмме дополнительную информацию извне через глобальные переменные, включаемые файлы и т.п. запрещается.

6. Класс "список" должен содержать подпрограмму вывода на экран списка длины не более r. Эта подпрограмма используется для вывода исходного списка после его инициализации, а также для вывода на экран результата. Подпрограмма выводит на экран не более, чем r элементов списка, где r – параметр этой подпрограммы (аргумент командной строки). Каждый элемент списка должен печататься на новой строке.

7. Класс "список" должен содержать функцию вычисления длины списка. Эта функция используется для вывода длины исходного списка после его инициализации, а также для вывода на экран длины результирующего списка.

8. Программа должна выводить на экран время, затраченное на решение.

Задачи

1. Написать подпрограмму – член класса "список", получающую в качестве аргумента целое число k и осуществляющую циклический сдвиг элементов списка на k позиций вправо.

2. Написать подпрограмму – член класса "список", получающую в качестве аргумента целое число k и выбрасывающую из него все элементы, большие какого-либо из k предыдущих элементов.

3. Написать подпрограмму – член класса "список", получающую в качестве аргумента целое число k и выбрасывающую из него все элементы, большие какого-либо из k следующих элементов.

4. Написать подпрограмму – член класса "список", получающую в качестве аргумента целое число k и выбрасывающую из него все k-локальные максимумы (элемент x_i назывется k-локальным максимумом, если $x_{i-k} \le \cdots \le x_{i-1} \le x_i \ge x_{i+1} \ge \cdots \ge x_{i+k}$).

5. Написать подпрограмму – член класса "список", получающую в качестве аргумента целое число k и выбрасывающую из него все участки постоянства, имеющие длину не менее k (участок удаляется целиком).

6. Написать подпрограмму – член класса "список", получающую в качестве аргумента целое число k и выбрасывающую из него все участки нестрого убывания, имеющие длину не менее k (участок удаляется целиком).

7. Написать подпрограмму – член класса "список", получающую в качестве аргумента целое число k и выбрасывающую из него все участки между участками постоянства, имеющими длину не менее k (участок удаляется целиком).

8. Написать подпрограмму – член класса "список", получающую в качестве аргумента целое число k и выбрасывающую из него все участки между участками нестрого возрастания, имеющие длину не менее k (участок удаляется целиком).