Требования к программам

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

```
class student
  {
  private:
    char * name;
    int    value;
  public:
    ...
  };
class list_node : public student
  {
  private:
    list_node * next;
  public:
    ...
  };
```

   Сам объект "список" в этом задании не используется, все функции получают указатель на начало списка.

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.

   Например, запуск

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

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

3. Ввод списка должен быть оформлен в виде подпрограммы, находящейся в отдельном файле.

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 и получающей в качестве аргументов указатель на начало списка. Получать в этой подпрограмме дополнительную информацию извне через глобальные переменные, включаемые файлы и т.п. запрещается.

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

Задачи

1. Написать функцию, получающую в качестве аргумента указатель на начало однонаправленного списка элементов типа `student` и возвращающую целое значение, равное количеству максимальных элементов этого списка.

2. Написать функцию, получающую в качестве аргумента указатель на начало однонаправленного списка элементов типа `student` и возвращающую целое значение, равное количеству элементов этого списка, больших предыдущего элемента.

3. Написать функцию, получающую в качестве аргумента указатель на начало однонаправленного списка элементов типа `student` и возвращающую целое значение, равное количеству локальных максимумов этого списка.

4. Написать функцию, получающую в качестве аргумента указатель на начало однонаправленного списка элементов типа `student` и возвращающую целое значение, равное 1, если этот список возрастает, 2, если убывает, 3, если постояннен, 4, если недостаточно данных для принятия решения, и 0, если не монотоннен.

5. Написать функцию, получающую в качестве аргумента указатель на начало однонаправленного списка элементов типа `student` и возвращающую целое значение, равное длине максимального участка строгого возрастания этого списка.

6. Написать функцию, получающую в качестве аргумента указатель на начало однонаправленного списка элементов типа `student` и возвращающую целое значение, равное количеству участков постоянства этого списка.

7. Написать функцию, получающую в качестве аргумента указатель на начало однонаправленного списка элементов типа `student` и возвращающую целое значение, равное максимальному расстоянию между соседними участками постоянства этого списка.

8. Написать функцию, получающую в качестве аргумента указатель на начало однонаправленного списка элементов типа `student` и возвращающую целое значение, равное длине максимального участка нестрогой монотонности этого списка.