Требования к программам

1. Программа работает с бинарным деревом объектов типа `student`:

```
class student
  {
  private:
    char * name;
    int    value;
  public:
    ...
  };
class tree_node : public student
  {
  private:
    tree_node * left;
    tree_node * right;
  public:
    ...
    friend class tree;
  };
class tree
  {
  private:
    tree_node * root;
  public:
    ...
    int read (const char * filename);
    void print (int r);
  };
```

   Все функции в задании являются членами класса "дерево".

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

   1) *r* – количество выводимых уровней в дереве,
   2) `filename` – имя файла, откуда надо прочитать дерево.

   Например, запуск

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

   означает, что дерево надо прочитать из файла `a.txt`, выводить не более 4-х уровней дерева.

3. Класс "дерево" должен содержать функцию ввода дерева из указанного файла.

4. Ввод дерева из файла. В указанном файле находится дерево 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. Длина слова неизвестна, память под него выделяется динамически. При заполнении первый объект типа `student` попадает в корень дерева, каждый новый объект типа `student` добавляется в левое поддерево, если он меньше текущего узла относительно операции сравнения `tree_node::operator<`, и в правое поддерево иначе. **Никакие другие функции, кроме функции ввода дерева, не используют упорядоченность дерева**. 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. Вывод результата работы функции в функции `main` должен производиться по формату:

   ```
   printf ("Task = %d Result = %d Elapsed = %.2f\n", task, res, t);
   ```

   где

   - `task` – номер задачи (1–5),
   - `res` – результат работы функции, реализующей решение этой задачи,
   - `t` – время работы функции, реализующей решение этой задачи.

   Вывод должен производиться в точности в таком формате, чтобы можно было автоматизировать обработку запуска многих тестов.

7. Класс "дерево" должен содержать подпрограмму вывода на экран не более чем *r* уровней дерева. Эта подпрограмма используется для вывода исходного дерева после его инициализации, а также для вывода на экран результата. Подпрограмма выводит на экран не более, чем *r* уровенй дерева, где *r* – параметр этой подпрограммы (аргумент командной строки). Каждый элемент дерева должен печататься на новой строке и так, чтобы структура дерева была понятна.

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

9. Поскольку дерево не изменяется функциями из задач, то для всех задач надо сделать **одну функцию** `main`, в которой прочитать дерево, вывести указанное число уровней на экран, и вызвать все функции задач, выводя результаты и время их работы. Другими словами, после компиляции должен получиться один исполняемый файл `a.out`, а не несколько.

Задачи

1. Написать подпрограмму – член класса "дерево", возвращающую целое значение, равное количеству концевых элементов этого дерева.

2. Написать подпрограмму – член класса "дерево", возвращающую целое значение, равное длине наибольшей ветви этого дерева.

3. Написать подпрограмму – член класса "дерево", возвращающую целое значение, равное максимальному количеству элементов в одном уровне этого дерева.

4. Написать подпрограмму – член класса "дерево", возвращающую целое значение, равное максимальной разности между глубинами левого и правого поддеревьев в узлах дерева.

5. Написать подпрограмму – член класса "дерево", возвращающую целое значение, равное количеству элементов, имеющих ровно 1 потомка.