Задание 1 – утилита linesort.

На 3. Считать со стандартного ввода строки и сформировать из них однонаправленный список. Отсортировать список методом пузырька (расположить строки в лексикографическом порядке). Вывести отсортированный список на стандартный вывод. Длина и количество строк не ограничены.
* Здесь под строкой (line) подразумевается последовательность символов до первого символа перевода строки '\n', но не включая его; другое значение слова строка (string) – массив символов, заканчивающийся символом с кодом 0.
* Для ввода строк описать функцию getline_unlim без параметров, возвращающую очередную считанную строку, либо NULL в случае конца файла. Считывание производить с помощью fgets в массив размером FGETS_BUFSIZE символов, где FGETS_BUFSIZE – константа, которую можно изменять без необходимости изменять код getline_unlim. При отладке использовать FGETS_BUFSIZE=5.
* Для сортировки описать функцию bubble_sort, принимающий список и возвращающий отсортированный методом пузырька список.
* Для вывода списка строк описать функцию print_list, для освобождения выделенной памяти – free_list.

На 4: то же, что на 3 плюс. Функция merge_sort – сортировка слиянием; выбор метода сортировки с помощью параметра командной строки (**-m** – merge sort, **-b** – bubble sort). Сравнить скорость работы реализованных методов на входных файлах различного размера (10^3, 10^4, 10^5, 10^6 строк). Для замера времени работы программы можно использовать утилиту time (см. man time). Для подготовки файлов с нужным количеством строк можно использовать утилиты cat и head. Запуски производить вручную, либо написать для этого bash-скрипт.

На 5: то же, что на 4 плюс. Утилита linesort_tree – аналог linesort, но считывает строки не в список, а в бинарное дерево поиска (не требуется сортировка). Сравнить скорость работы linesort и linesort_tree; для этого написать bash-скрипт, который скачивает необходимые для тестирования скорости файлы (список URL должен быть в отдельном файле), из них формирует входные файлы необходимого размера (см. выше), сортирует их с помощью linesort –m, linesort –b, linesort_tree и выводит время каждого прогона.

Задание 2 – построение частотного словаря.

Для выполнения задания понадобится утилита сортировки строк со стандартного ввода. Если есть свои утилиты linesort/linesort_tree, необходимо использовать их; если нет – воспользоваться стандартной утилитой sort.

На 3. Выделить функцию getline_unlim из задания 1 в отдельный модуль util (если задание 1 не выполнялось, необходимо описать такую функцию); для этого создать заголовочный файл **util.h** с прототипом функции и **util.c** с ее описанием. Утилиты **wordsplit** и **countd** необходимо оформить как многомодульные программы; файлы **wordsplit.c** и **countd.c** должны подключать заголовочный файл **util.h** с помощью директивы #include и использовать функцию getline_unlim. В модуль util можно добавить другие полезные функции (по вкусу).

Утилита **wordsplit**: разбить текст на стандартном вводе на слова, при наличии флага **–l** конвертировать слова в нижний регистр (VaSYa→vasya), вывести на стандартный вывод по одному слову в строке.

Утилита **countd**: считать строки со стандартного ввода и вывести на стандартный вывод; повторяющиеся подряд идущие строки выдать 1 раз; слева от каждой строки вывести количество повторов. Формат вывода:

<количество повторов, выровненное по правому краю поля шириной 4 символа><пробел><исходная строка>

С помощью написанных утилит и операторов шелла '|', '<', '>' сформировать частотный словарь – файл **dict.txt**, в котором для любого слова указано число его вхождений в исходный текст (пример – справа). Исходный текст сформировать, скачав с помощью утилиты wget как минимум 10 англоязычных текстов и объединив их с помощью утилиты cat (см. Ценные указания).

```
   4 wisdom
   1 wise
   3 wish
   2 wished
   1 wishes
   1 wit
 138 with
   9 within
  21 without
   1 witnesses
   1 wonderful
```

* Для разбиения на слова достаточно заменить все пробельные символы и знаки препинания на символ перевода строки.
* Сборку реализовать с помощью утилиты make: создать файл **Makefile** с целями all (сборка обеих утилит), wordsplit (сборка утилиты wordsplit), countd (сборка утилиты countd), util.o (сборка объектного файла util.o); прописать зависимости между целями, зависимости целей от файлов, команды сборки целей.

* Утилита make описана в http://www.stolyarov.info/books/pdf/unixref.pdf (параграф 4.3) и https://ejudge.ru/study/3sem/makefile.pdf (читать оба файла!).
На 4. то же, что на 3 плюс. Изучить стандартные утилиты tr, uniq и поддерживаемые ими флаги (man tr, man uniq); предложить решение задачи формирования частотного словаря, использующее только стандартные утилиты sort, tr, uniq. Сравнить производительность двух полученных решений на исходных текстах различного размера.
Изучить флаги -r, -k, -n утилиты sort. Каким образом можно упорядочить словарь по числу вхождений слова в текст (начиная с наиболее частотных)?
* Добавить в Makefile цели data (скачивает указанные в books файлы и объединяет их содержимое в один файл data), dict_std.txt (строит частотный словарь с помощью стандартных утилит, выводит затраченное время), dict.txt (строит частотный словарь с помощью созданных утилит, выводит затраченное время). Добавить необходимые зависимости, чтобы при изменении файла books (например, добавлении новой ссылки) цели dict_std.txt и dict.txt становились устаревшими и словари перестраивались.
* Добавить цель test, которая запускает оба способа построения частотного словаря и выводит затраченное время, чтобы можно было сравнить производительность.
* Добавить цель sorted_dict.txt, которая строит словарь, слова в котором упорядочены по частоте (начиная с наиболее частотных).
На 5. то же, что на 4 плюс. Утилита **freqdict**, строящая частотный словарь более эффективным способом – без сортировки. Утилита считывает строки со стандартного ввода, разбивает их на слова и заносит слова в ассоциативный массив (ключ – слово, значение – число вхождений). Ассоциативный массив реализовать с помощью хэш-таблицы в отдельном модуле hashtable.
Сравнить производительность утилиты freqdict и двух предложенных ранее решений (добавить и модифицировать соответствующие цели).

Ценные указания.

* Проект http://www.gutenberg.org/ позволяет скачивать книги в формате plain text UTF-8 (текстовый файл в кодировке UTF-8). Их можно использовать в качестве данных для тестирования. Кодировка UTF-8 использует от 1 до 4 байт для кодирования одного символа (в зависимости от символа) и включает кодировку ASCII в качестве подмножества. Это означает, что символы английского алфавита, цифры, основные знаки пунктуации и пробельные символы в UTF-8 кодируются 1 байтом, причем их коды в UTF-8 совпадают с их кодами в ASCII. В связи с этим, желательно использовать для тестирования англоязычные тексты – они будут корректно обрабатываться изученными средствами языка Си.

* Чтобы подать несколько файлов на стандартный ввод любой утилиты можно использовать cat (последовательно выводит содержимое всех файлов, заданных в качестве аргументов, на стандартный вывод), например:
cat f1 f2 f3 | sort
Вместо имен файлов можно указывать шаблон имени, интерпретатор командной строки автоматически подставит имена, соответствующие шаблону:
cat *txt | sort – подаст на стандартный ввод sort содержимое всех файлов в текущей директории, с именами, оканчивающимися на “txt”.

* Скачать файл можно с помощью утилиты wget, в качестве аргумента необходимо указать URL файла (его можно скопировать из адресной строки после открытия файла в браузере), например:
wget http://www.gutenberg.org/ebooks/47067.txt.utf-8
Чтобы скачать файлы по списку URL, занесенному в текстовый файл books, достаточно выполнить:
wget `cat books`
Шелл выполняет команду в обратных кавычках, записывает ее стандартный вывод в строку и подставляет эту строку вместо команды в обратных кавычках.

* Оценить количество строк в файле можно с помощью утилиты wc. Объединить несколько файлов в один – с помощью cat. Взять нужное количество строк – с помощью head.

* чтобы скопировать текст из командной строки, достаточно его выделить – текст автоматически помещается в буфер обмена; чтобы вставить текст в текущую позицию курсора в командной строке, нужно нажать правую кнопку мыши, либо Shift+Insert

* fgets возвращает NULL либо в случае конца файла (при условии, что ничего не было прочитано), либо в случае ошибки. Чтобы различать эти случаи, можно использовать функцию feof. В случае ошибки нужно вывести на стандартный поток вывода ошибок сообщение, содержащее информацию о месте и причине возникновения ошибки, и немедленно завершить программу с ненулевым кодом возврата (функция exit с параметром EXIT_FAILURE) . Функция fgets записывает код ошибки в глобальную переменную errno – к этой переменной можно обратиться, если подключить заголовочный файл errno.h. Для получения по коду ошибки строки с описанием этой ошибки служит функция strerror. Вывод на стандартный поток вывода ошибок можно осуществлять с помощью функции fprintf, передав ей в качестве потока вывода константу stderr.

* получить справку по любой библиотечной функции можно с помощью утилиты man; например:

```
$ man fgets
```

В некоторых случаях имя библиотечной функции совпадает с именем встроенной команды шелла, либо утилиты. Например, по запросу "man exit" выдается справка по встроенной команде шелла с именем "exit" (раздел 1 справки). Чтобы получить справку по одноименной библиотечной функции Си (из раздела 3 справки), необходимо явно указать необходимый раздел:

```
$ man 3 exit
```

Внимательно и полностью читайте справку по всем используемым библиотечным функциям – в ней часто содержится важная информация о нюансах работы и рекомендации по использованию функции.

* во многих утилитах коммандной строки (man, less, vim, …) поиск в текущем файле осуществляется так:

а) нажмите клавишу "/" (в vim необходимо делать это в командном режиме)

б) введите искомую строку и нажмите Enter

в) с помощью клавиш "n" и "Shift+n" можно перемещаться к следующему и предыдущему вхождению искомой строки

* некоторые полезные флаги ls, которые делают ее вывод намного полнее и читабельнее: -l, -h, -a, -t (читайте про них в man), флаги можно комбинировать: "ls -lhat"

* если командная строка все еще доставляет вам дискомфорт, избавиться от него помогут следующие материалы:

http://www.ee.surrey.ac.uk/Teaching/Unix/

http://www.stolyarov.info/books/pdf/unixref.pdf