Завдання 1:

Необхідно реалізувати простий аналог *memcached* серевера, який приймає наступні команди:

- get (key);
- set (key, value, expires?) (з можливістю, опціонально, задати період зберігання записів у секундах, по замовчуванню період зберігання не обмежений)
- delete (key).

Сервер повинен вміти обслуговувати одночасно кількох клієнтів (паралельно). У одній сесії клієнт може виконувати кілька запитів. Дані зберігати у пам'яті та у файлі, доступ до якого відбувається через *mmap*. По сигналу USR1 створювати файл у **/tmp** зі списком записів ('key' -> 'value'). Протокол — текстовий (можна зробити повністю сумісним із *memcached)*.

Платформа - POSIX (FreeBSD/Linux). Використання посторонніх бібліотек дозволяється. Бажано по можливості використовувати Стандартну Бібліотеку C++ (**C++11**). Є небажаним заміна компонентів, які існують у Стандартній Бібліотеці компонентами з інших бібліотек (напр. *std::thread* на *boost::thread*, *std::vector<std::unique_ptr>* на *boost::ptr_vector*). Додаткові бали за абстрагування прошарку збереження даних (для можливості легкої зміни сховища даних – файл, БД, ітд.), за абстрагування моделі обробки з'єднань (напр: fork / pre-fork / threaded / event-based) від роботи з текстовим протоколом.

Завдання 2:

Підзавдання 1: Є вхідний файл зі списком слів у кодуванні UTF-8 (кількість слів невелика, можливе завантаження всього списку у пам'ять). Вміст вхідного файлу ніколи не змінюється. Періодично запускається програма, яка отримує на вході декілька слів і видає ті з них, які присутні у цьому списку. Слова у файлі, на вході програми і на її виході розділені новим рядком. Потрібно реалізувати алгоритм знаходження слова у списку, який не вимагає постійного завантаження вхідного файлу у пам'ять і який працює ефективніше ніж O(n).

Підзавдання 2: Є список чисел, елементи якого можуть додаватись та видалятись. Числа унікальні. До списку виконуються запити на вибірку 'сторінки' чисел. Сторінка - це діапазон чисел, які ідуть послідовно один за одним і є відсортовані у спадному порядку (перший елемент першої сторінки – найбільше число у списку). Сторінки мають фіксований розмір (наприклад 25, 50, 100, …). Кількість елементів на останній сторінці може бути меншою за розмір сторінки. Необхідно реалізувати структуру даних списку на диску і алгоритм вставки елемента у список, видалення елемента зі списку та вибірки довільної сторінки, який працює ефективніше за O(n).