Контрольная работа № 2. ООП. Линейные структуры данных. Работа с памятью.
Вариант 3

Указания:

1. *Решение каждой из задач следует оформить в виде отдельного файла с именем вида* ***kr2_gr3b_v1_ivanov_ivan.zip. (в ***.**zip* ***архиве Вы поместите файлы исходного кода + input.txt).***
2. *Оценка за контрольную равняется min(10, round(0,5 + x)), где 0.5 - бонус, x - сумма набранных по задачам баллов (10.5 баллов можно набрать выполнив все пункты контрольной работы).*
3. *Для получения полного балла, решение всех подзадач в задании необходимо оформить в виде класса(классов) с членами-функций. Таким образом, в функции main не должно быть никакой бизнес-логики, только переменные с данными и вызовы методов для решения подзадач.*
4. *При реализации алгоритмов не используйте стандартные контейнеры, а используйте собственную реализацию контейнеров.*

Задание:
Разработать объектно-ориентированное приложение приложения для решения задачи:

Дан текстовый файл *input.txt*, содержащий количество чисел (n, где 2≤n≤100000) и последовательность целых неотрицательных чисел (все числа меньше 100000).

Необходимо сложить все числа так, чтобы суммарная стоимость их сложения была наименьшая (“***стоимость***” операции сложения чисел a и b равна их сумме: a+b).

Указание: Стоимость сложения всех чисел будет минимальной, если на каждом следующем шаге складывать два наименьших числа из множества, состоящего из данного ряда чисел и уже вычисленных «частичных сумм». Для решения задачи используйте очередь с приоритетом, наибольшим приоритетом обладает число с минимальным значением.

Пример:

input.txt	output.txt
3 3 1 2	9
4 3 2 4 1	19

Определение: Предположим, что для каждого элемента определен некоторый приоритет. В простейшем случае значение приоритета может совпадать со значением элемента. В общем случае соотношение элемента и приоритета

может быть произвольным. **Приоритетной очередью** (англ. priority queue) называется такая *абстрактная структура данных*, интерфейс которой включает в себя следующие операции:
1) Pull() поиск и удаление (~возвращение) элемента с самым высоким приоритетом;
2) Insert(x; prior(x)) добавление элемента x с указанным приоритетом.
Интерфейс структуры данных “приоритетная очередь” может быть реализован на основе различных структур данных

1. **(1 балл)** Реализовать класс Queue (и NodeQueue), реализующий интерфейс очереди на последовательности связанных компонент. Контруктор(ы), деструктор.
2. **(0.5 балла)** Элемент коллекции (NodeQueue) реализован через внутренний класс.
3. **(2.5 балла)** Реализовать класс PriorityQueue (и NodePriorityQueue), реализующий интерфейс “приоритетной очереди” на последовательности связанных компонент - на основе списка, в котором элементы упорядочены по убыванию приоритета элементов. Конструктор(ы), деструктор.
4. **(0.5 балла)** Элемент коллекции (NodePriorityQueue) реализован через внутренний класс.
5. **(0.5 балла)** Классы Queue, PriorityQueue и NodeQueue, NodePriorityQueue реализованы через шаблоны.
6. **(1 балла)** Функция **ReadInput**, которая принимает в качестве аргумента имя входного файла, ссылку на коллекцию-очередь (Queue) элементов, значение которых имеет числовой тип (или Element, см.п.7). Числа читаются из файла, [помещаются в объект класса Element (см.п.7)], и добавляются в очередь.
7. **(0.5 балла)** Если для числа будет создан класс-обертка Element, который в своем поле хранит число, в NodeQueue поле value имеет тип Element, т.е. числа исходной последовательности помещаются в объекты класса Element.
8. **(2 балл)** Функция **Convertion**, которая имеет единственный аргумент: ссылку на коллекцию, созданную на основе исходных данных (очередь), и возвращает коллекцию - “приоритетную очередь”, построенной с учетом того, что наибольший приоритет имеет число с наименьшим значением (приоритет числа a равен -a).
9. **(0.5 балл)** Функция **Calculation**, которая имеет единственный аргумент: ссылку на “приоритетную очередь” (п.8), и возвращает число – стоимость операций сложения.
10. **(0.5 балла)** Функция **WriteCollection**, которая выводит полученный результат (имя файла - параметр).
11. **(0.5 балла)** Проверить корректность вызовов и передаваемых в методы параметров с помощью assertов.