**Лабораторная работа 7.**

**Задание 1. Алгоритмы STL**

В соответствии с Вашим вариантом нужно:

1. Реализовать функцию, которая принимает на вход ссылку на объект string и заменяет числа в строке следующим образом: если это число от 0 до 6, то заменить его на названия дня недели соответственно (0 – Sunday, 1 – Monday и т.д.); если это число от 10 до 100, то записать их прописью. (20 – twenty, 45 – forty-five и т.п.)

   Пример: «1: 15 dollars» -> «Monday: fifteen dollars».

2. Реализовать функцию, которая принимает на вход ссылку на объект string и заменяет названия дней неделей (независимо от регистра) на соответствующие следующим образом: Sunday – 0, Monday – 1 и т.д.; а если это числа, записанные прописью, то заменить их на соответствующие числа от 10 до 100. (twenty – 20, forty-five – 45 и т.п.)

   Пример: «Monday: fifteen dollars» -> «1: 15 dollars».

3. Реализовать функцию, которая принимает на вход ссылку на объект string и заменяет строку на строку следующего вида: «N palindromes», где N – количество чисел палиндромов, встречающихся в исходной строке.

   Пример: «3443 and 616 are palindromes, but 145 is not» -> «2 palindromes».

4. Реализовать функцию, которая принимает на вход ссылку на объект string и сначала сортирует все числа в порядке неубывания, а затем заменяет числа на их остаток по модулю 3.

   Пример: «16 Ivanov 14 Petrov 15 Sidorov» -> «2 Ivanov 0 Petrov 1 Sidorov».

5. Реализовать функцию, которая принимает на вход ссылку на объект string и в зависимости от длины строки делает следующее:

   а) Если длина чётная, то вернуть отсортированный список чисел в строке, являющихся целыми степенями двойки. Пример: «This 2 22 string has 4 2 1 even 128 129 length» -> 5.

   б) Если длина нечётна, то вернуть отсортированный список чисел в строке, НЕ являющихся целыми степенями двойки. Пример: «This 2 22 string 153 has 4 2 1 odd 128 129 length» -> 3.

6. Реализовать функцию, которая принимает на вход ссылку на объект vector<int> и меняет местами чётные и нечётные числа в порядке их появления. Если одних чисел больше, то их оставшуюся часть (которую уже не с чем менять) заменить на -1. Полученный список отсортировать в порядке невозрастания.

   Пример: {1, 2, 2, 3, 4, 5, 2, 3, 1, 1337, 2, 4, 6, 6, 6} -> {2, 1, 3, 2, 5, 4, 3, 2, 2, 4, 1 1337, -1, -1, -1}.
   ( {1, 1337, 2, 4} -> {2, 4, 1, 1337} – 1 <-> 2, 1337 <-> 4 )

7. Реализовать функцию, которая принимает на вход ссылку на объект vector<int> и сортирует все чётные числа в порядке неубывания (после сортировки числа занимают места до сортировки), а затем убирает все простые числа из полученного вектора.

   Пример: {1, 1, 22, 23, 17, 2022, 144, 1337, 2} -> {1, 1, 2, 23, 17, 22, 144, 1337, 2022} ->
   -> {1, 1, 22, 144, 1337, 2022}

8. Реализовать функцию, которая принимает на вход ссылку на объект vector<string> и сортирует все строки в лексикографическом порядке, после чего убирает из них все гласные и соединяет полученные строки в одну (её же и возвращать).

Пример: {“just”, “an”, “example”} -> {“an”, “example”, “just”} -> {“n”, “xmpl”, “jst”} ->
-> “nxmpljst”.

9. Реализовать функцию, которая принимает на вход ссылку на объект map<int, string> и возвращает строку, в которой для каждого ключа N строка S повторяется N%2 + 1 раз (сама строка должна быть отсортирована в лексикографическом порядке), порядок таких строк должен соответствовать неубывающему порядку ключей.

Пример: {1: “number one”, 1337: “leet”, 2022: “year”}->“ beemnnoru beemnnorueelteeltaery”.
(“number one” -> “ beemnnoru” – 1%2+1 раз = 2, “leet” -> “eelt” – 1337%2+1 раз = 2, “year” -> “aery” – 2022%2+1 = 1)

10. Реализовать функцию, которая принимает на вход ссылку на объект string и каждую **цифру** дублирует число раз, равное этой цифру.

Пример: “1 2 string 3 4 is 5 6 nice” -> “1 22 string 333 4444 is 55555 666666 nice”.

11. Реализовать функцию, которая принимает на вход ссылку на объект vector<int>, находит в нём медианный элемент и все числа больше него заменяет на максимум множества, а меньше него – на минимум.

Пример: {15, 213, 1, 27, 2, 1337, 2022} -> {1, 1, 1, 27, 2022, 2022, 2022}.

12. Реализовать функцию, которая принимает на вход ссылку на объект vector<double>, заменяет каждое число на его целую часть, а дробную часть дописывает в конец вектора, после чего сортирует данное множество.

Пример: {4.15, 2.13, 1.01, 23.99, 22.01} -> {4, 2, 1, 23, 22, .15, .13, .01, .99, .01} ->
-> {.01, .01, .13, .15, .99, 1, 2, 4, 22, 23}.

13. Реализовать функцию, которая принимает на вход ссылку на объект queue<int> и удаляет из очереди все простые числа, а после все целочисленные степени двойки перемещает в конец очереди в отсортированном порядке.

Пример: {1, 1, 2, 12, 13, 22, 32, 1337, 256, 64, 2022} -> {1, 1, 12, 22, 32, 1337, 256, 64, 2022} ->
-> {1, 1, 12, 22, 1337, 2022, 32, 256, 64}.

14. Реализовать функцию, которая принимает на вход ссылку на объект vector<set<int>> и возвращает пересечение всех множеств вектора (сетов), если оно не пустое, а если пустое, то возвращает их объединение.

Пример: {{1, 2, 3}, {2, 3, 14}, {3, 14, 15}, {3, 1337}} -> {3}.

**Задание 2. Контейнеры.**

1. Реализовать функцию, которая принимает на вход: vector<set<int>> &vec и vector<int> &res и генерирует массив целых чисел res из массива множеств целых чисел vec по следующим правилам: res содержит только числа из множеств, не содержащих отрицательные числа; числа из одного множества отсортированы в порядке неубывания; числа в итоговом массиве должны быть распложены так, что по порядку сначала идут числа из того множества, в котором есть максимальное число из других множеств (например, из множеств A = {1, 2, **3**}, B = {4, 5, **6**} и C = {7, 8, **9**}, в итоговый массив сначала будут взяты числа из множества C, затем из множества B, затем из множества A). При совпадении максимумов порядок любой.

Например: vec = {{-3,8,11}, {2,6,4,5}, {1,0,7,5},{1,3,2}} => res = {0,1,5,7,2,4,5,6,1,2,3}

2. Реализовать функцию, которая принимает на вход: v vector<list<int>> &vec и vector<int> &res и генерирует массив целых чисел res из массива списков целых чисел vec по следующим правилам: res содержит только числа из списков, содержащих менее шести элементов; числа, принадлежащие одному списку, отсортированы в порядке невозрастания; числа, принадлежащие разным спискам отсортированы в порядке возрастания среднего арифметического списка (например, из списков A = {6, 10}, B = {4, 4} и C = {2, 8}, в итоговый массив сначала будут взяты числа из множества B (среднее арифметическое = **4**), затем из множества C (среднее арифметическое = **5**), затем из множества A (среднее арифметическое = **8**).

Например: vec ={{9,22,10,2,6,3,9,1}, {4,6}, {8,10}} => res = {6,4,10,8}

3. Реализовать функцию, которая принимает на вход: string &str, vector<set<string>> &res, которая формирует массив множеств строк res из строки str по следующему правилу: res[i] – это множество слов из строки str, количество гласных букв в которых равно i. Слова разделены пробелом.

Например: str = ”canst thou o cluel say I love thee not when I against myself with thee partake” => res = {{},{“canst”, “o”, “I”, “not”, “when”, “with”},{“thou”, “cluel”, “say”, “love”, “thee”, “myself”},{“against”,” partake”}}

4. Реализовать функцию, которая принимает на вход: vector<string> &vec, map <string, int> &res, которая формирует из массива кодов цвета ассоциативный массив следующего вида: ключом является название цвета радуги (красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый), а значением – количество цветов, соответствующих данной группе. Таким образом, следует сгруппировать оттенки цветов по цветам радуги.

Например: vec = {“ #008000”, “#FF0000”, “#228B22”, “#B22222”, “#FF8C00”, “#800080“} -> res = {“red”: 2; “orange”: 1; “green”: 2; “purple”: 1, “yellow”: 0 …}

5. Реализовать функцию, которая принимает на вход: vector<vector<string>> &vec, map<int, string> &res, которая подсчитывает вхождение каждого слова, а затем заполняет ассоциативный массив следующим образом: для строки S ключом становится значение N*Sum(I)*Sum(J) – где N – количество вхождений данной строки во все подмассивы vec, Sum(I) – сумма всех индексов множеств, где есть данная строка, Sum(J) – сумма всех индексов строки в подмножестве, где она есть (индексацию считать с 1). При коллизии пар ключ-значение сделать значением строку, меньшую по лексикографическому порядку.

Например: vec = {{“example”, “string”, “nothing”}, {“string”, “is”, “nice”}, {“empty”, “vector”}} -> res = {1: “example”, 2: “vector”, 3: “empty”, 4: “is”, 6: ”nice”, 18: “string”} («empty» и «nothing» оба с ключом 3, но «empty» меньше)

6. Реализовать функцию, которая принимает на вход: vector<pair<int, int>> &vec, vector<double> &pts такие, что каждая пара в vec – это два индекса от 0 до pts.size() – 1. Соответственно, требуется получить массив, в котором эти индексы заменены на сами точки из массива pts, а после отсортировать его по модулю разности между значениями в паре, причём в самой паре меньшее число должно быть первым.

Например: vec = {{0, 1}, {1, 3}, {2, 3}}, pts = {1.68, 2.71, 3.14, 5.15} -> {{1.68, 2.71}, {2.71, 5.15}, {3.14, 5.15}} -> {{1.68, 2.71}, {3.14, 5.15}, {2.71, 5.15}}.

7. Реализовать функцию, которая принимает на вход: multiset<int> &st1, multiset<int> &st2. Из обоих мультимножеств требуется убрать элементы, степень вхождения которых больше, чем само это число, а затем получить массив пар, где первый элемент – это максимальная степень вхождения числа одновременно в оба мультимножества, а второй – само число и отсортировать данный массив по значению чисел (второй элемент).

Например: st1 = {1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5}, st2 = {1, 2, 2, 3, 3, 3, 3, 5, 15} ->
-> st1 = {1, 3, 3, 3, 5}, st2 = {1, 2, 2, 5, 15} -> {{1, 1}, {2, 2}, {0, 3}, {1, 5}, {0, 15}}

8. Реализовать функцию, которая принимает на вход: multiset<double> &st1, multiset<int> &st2 равных размеров. Требуется оставить во втором мультимножестве только те числа, которые отличаются не больше, чем на 2 от дробной части (после точки) умноженной на 10 соответствующего числа из первого мультимножества (соответствие по индексам). А затем найти разность полученного множества и изначального второго мультимножества.

Например: st1 = {1.68, 2.71, 3.14}, st2 = {7, 5, 3} -> {7, 3} -> {3, 7} -> {5}

9. Реализовать функцию, которая принимает на вход: vector<pair<int, int>> &vec1, vector<pair<int, int>> &vec2. Требуется получить новый массив вида vector<pair<pair<int, int>, int>>, в котором каждая пара (перебираются всевозможные пары) содержит ещё одну пару – индексы пар из первого и второго массива, а вторым элементом их скалярное произведение (как если бы это были алгебраические векторы). Данный массив нужно отсортировать по неубыванию скалярного произведения.

Например: vec1 = {{1, 2}, {2, 3}}, vec2 = {{0, 10}, {2, 20}, {-1, 0}} -> {{{0, 0}, 20}, {{0, 1}, 42}, {{0, 2}, -1}, {{1, 0}, 30}, {{1, 1}, 64}, {{1, 2}, -2}} -> {{{1, 2}, -2}, {{0, 2}, -1}, {{0, 0}, 20}, {{1, 0}, 30}, {{0, 1}, 42}, {{1, 1}, 64}}

10. Реализовать функцию, которая принимает на вход: multimap<int, string> mp1, map<int, int> mp2, vector<pair<string, int>> vec. Целочисленные значения в vec1 соответствуют ключам из mp2, а значения mp2 соответствуют ключам mp1. Требуется оставить в vec1 только те пары, в которых целочисленное значение приводит к строке (хотя бы одной) из mp1, равной строке из пары, а затем отсортировать полученные пары по неубыванию целочисленного значения.

Например: mp1 = {1: “string”, 2:”not string”, 3: “null”}, mp2 = {15: 1, 21: 2, 33: 3}, vec = {{“string”, 15}, {“nothing”, 33}, {“not string”, 21}} -> {{“string”, 15}, {“not string”, 21}}.

11. Реализовать функцию, которая принимает на вход: vector<vector<int>> &M, vector<int> &vec, причём все числа в M от 0 до vec.size() – 1. Каждое значение в M – это индекс значения в vec. Требуется отсортировать столбцы (не строки) в M так, чтобы знакочередующаяся сумма их элементов (элемента из vec по индексу из M) шла от меньшей к большей (слева направо). Необходимо вернуть map<int, pair<int, vector<int>> - где ключ – индекс столбца, первое значение пары – его знакочередующаяся сумма, а второе – сами значения столбца.

Например: M = {{0, 1, 2},
{2, 1, 0},
{1, 0, 2}}
vec = {1, 12, 42} ->

M = {{1, 12, 42},
{42, 12, 1}, знакочередующаяся сумма по столбцам: -29, 1, 83
{12, 1, 42}}

12. Реализовать функцию, которая принимает на вход: multiset<double> &st1, set<double> &st2, причём во втором множестве все числа входят в полуинтервал [0, 1). Необходимо оставить в мультимножестве только те числа, дробная часть которых есть в st2, а затем вернуть multimap<double, int>, где ключ – дробная часть числа, а значение – индекс числа с этой дробной частью в полученном мультимножестве.

Например: st1 = {1.68, 2.71, 3.14, 13.37}, st2 = {0.37, 0.71} -> st1 = {2.71, 13.37} ->
-> {0.37: 1, 0.71: 0}.

13. Реализовать функцию, которая принимает на вход: multimap<int, string> &mp1, vector<string> &vec, после чего достаёт все значения из mp1 и находит пересечение с массивом vec, а результат возвращает в виде multiset<string>, где каждая строка повторяется число раз, равное её ключу в mp1.

Например: mp1 = {1: “string”, 2: “not string”, 3: “null”}, vec = {“string”, “null”} ->
-> {“null”, “null”, “null”, “string”}.

14. Реализовать функцию, которая принимает на вход: multimap<int, string> &mp1, multimap<string, int> &mp2 и возвращает set<pair<int, string>>. Если по ключу N из mp1 получается значение S и по этому значению S в качестве ключа получается значение N из mp2 (тот же самый инт), то такую пару надо добавить в set<pair<string, int>>, но только, если по ключу N значение уникально (ровно одна строка). После вернуть это множество.

Например: mp1 = {1: “string”, 2: “not string”, 3: “null”, 4: “empty”, 4: “clear”}, mp2 = {“string”: 1, “null”: 2, “not string”: 2, “clear”: 4} -> {{1, “string”}, {2, “not string”}} (“clear” не подходит, так как по ключу 4 это не уникальная строка).

**Правила сдачи работы.**

Защищать работу можно во время пары Вашей группы и подгруппы. При защите следует продемонстрировать работу Вашей программы. По работе будут заданы вопросы на понимание кода и теоретические аспекты выполненного задания.

Примерные критерии оценивания за каждое из 2х заданий:

| **Баллы** | **Описание** |
|---|---|
| 0 | Задание выполнено неверно или(!) при защите было отмечено, что студент не разбирается в собственном коде, не дает очевидных ответов на вопрос по своему же коду, не дает ответов на элементарные вопросы по теории. **Обратите внимание, при обнаружении двух сильно похожих работ 0 ставится за ОБЕ работы.** |
| 1 | В выполнении задания есть ошибки. При защите практически на все вопросы не был дан правильный ответ. |
| 2 | В выполнении задания есть ошибки. При защите даны ответы не на все поставленные вопросы. |
| 3 | Задание выполнено верно, в коде могут быть недочеты. При защите даны ответы не на все поставленные вопросы. |
| 4 | Задание выполнено верно. При защите даны ответы на все поставленные вопросы. |
| 4.5 | Задание выполнено полностью верно. Студент показал блестящие теоретические знания и практические навыки, ответил на все поставленные вопросы. |

За данную лабораторную работу студент может получить 10 баллов, если он получил оценку 9 по сумме баллов + «звездочка» от лектора или семинариста.