|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/05 Современные интеллектуальные**

**программно-аппаратные комплексы**

**ОТЧЕТ**

**по лабораторной работе №7**

|  |  |
| --- | --- |
| Группа | ИУ6-23М |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | И.И. Иванов |
|  |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |
| Преподаватель |  |  | П.В. Степанов |
|  |  | (Подпись, дата) | (И.О. Фамилия) |

*Москва, 2023 г.*

**Условие**

1. Из текста удалить все слова заданной длины, начинающиеся на согласную букву.

2. Удалить из текста его часть, заключенную между двумя символами, которые вводятся (например, между скобками ‘(’ и ‘)’ или между звездочками ‘\*’ и т.п.).

3. В стихотворении найти одинаковые буквы, которые встречаются во всех словах.  
4. В тексте найти первую подстроку максимальной длины, не содержащую букв.

5. Напечатать слова русского текста в алфавитном порядке по первой букве. Слова, начинающиеся с новой буквы, печатать с красной строки.

6. Рассортировать слова русского текста по возрастанию доли гласных букв (отношение количества гласных к общему количеству букв в слове)

7. Преобразовать каждое слово в тексте, удалив из него все последующие (предыдущие) вхождения первой (последней) буквы этого слова

8. Исключить из текста подстроку максимальной длины, начинающуюся и заканчивающуюся одним и тем же символом

**Програмная реализация**

1

|  |
| --- |
| public class Main {  public static void main(String[] args) {  String text = "The quick brown fox jumps over the lazy and cool dog";  int n = 4;  String result = removeWords(text, n);  System.out.println(result);  }  public static String removeWords(String text, int n) {  String[] words = text.split("\\s+");  StringBuilder result = new StringBuilder();  for (String word : words) {  if (word.length() != n || isVowel(word.charAt(0))) {  result.append(word).append(" ");  }  }  return result.toString();  }  private static boolean isVowel(char c) {  return "AEIOUaeiou".indexOf(c) != -1;  }  } |

2

|  |
| --- |
| public class Main {  public static void main(String[] args) {  String text = "Это (текст), который нужно 'редактировать'.";  String editedText = removeBetween(text, args[0].charAt(0), args[1].charAt(0));  System.out.println(editedText); // Output: "Это , который нужно \*редактировать\*."  }  public static String removeBetween(String text, char start, char end) {  StringBuilder result = new StringBuilder();  int i = 0;  while (i < text.length()) {  if (text.charAt(i) == start) {  i++;  while (i < text.length() && text.charAt(i) != end) {  i++;  }  i++;  } else {  result.append(text.charAt(i));  i++;  }  }  return result.toString();  }  } |

3

|  |
| --- |
| import java.util.ArrayList;  import java.util.HashSet;  public class Main {  public static void main(String[] args) {  String text = "this test text";  String[] words = text.split("\\s+");  ArrayList<Character> result = new ArrayList<>();  for (char c = 'a'; c <= 'z'; c++) {  boolean found = true;  for (String word : words) {  if (word.indexOf(c) == -1) {  found = false;  break;  }  }  if (found) {  result.add(c);  }  }  System.out.println(result);  }  } |

4

|  |
| --- |
| import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class Main {  public static void main(String[] args) {  String text = "This is a test text 1234567890 !@#$%^&\*()\_+?><";  Pattern pattern = Pattern.compile("[^a-zA-Z]+");  Matcher matcher = pattern.matcher(text);  int maxLength = 0;  String longestSubstring = "";  while (matcher.find()) {  int start = matcher.start();  int end = matcher.end();  if ((end - start) > maxLength) {  maxLength = (end - start);  longestSubstring = text.substring(start, end);  }  }  System.out.println(longestSubstring);  }  } |

5

|  |
| --- |
| import java.util.Arrays;  public class Main {  public static void main(String[] args) {  String text = "в этом предложении мы должны отсортировать все слова по алфавиту начиная с первой буквы и дальше";  // Разбиваем текст на слова  String[] words = text.split("\\s+");  // Сортируем слова в алфавитном порядке по первой букве  Arrays.sort(words, (a, b) -> a.charAt(0) - b.charAt(0));  // Печатаем слова, начинающиеся с каждой буквы  char currentLetter = ' ';  for (String word : words) {  char firstLetter = word.charAt(0);  if (firstLetter != currentLetter) {  currentLetter = firstLetter;  System.out.println();  System.out.print(" ");  }  System.out.print(word + " ");  }  }  } |

6

|  |
| --- |
| import java.util.\*;  public class Main {  public static void main(String[] args) {  String text = "Рассортировать слова русского текста по возрастанию доли гласных букв";  String[] words = text.split("\\s+");  Map<String, Double> wordVowelRatios = new HashMap<>();  for (String word : words) {  int vowelCount = countVowels(word);  int letterCount = word.length();  double ratio = (double) vowelCount / letterCount;  wordVowelRatios.put(word, ratio);  }  // Sort words by vowel ratio ascending  List<Map.Entry<String, Double>> sortedWords = new LinkedList<>(wordVowelRatios.entrySet());  Collections.sort(sortedWords, Comparator.comparing(Map.Entry::getValue));  // Output sorted words  for (Map.Entry<String, Double> entry : sortedWords) {  System.out.println(entry.getKey());  }  }  private static int countVowels(String word) {  int count = 0;  for (char c : word.toLowerCase().toCharArray()) {  if (c == 'а' || c == 'е' || c == 'ё' || c == 'и' || c == 'о' || c == 'у' || c == 'ы' || c == 'э' || c == 'ю' || c == 'я') {  count++;  }  }  return count;  }  } |

7

|  |
| --- |
| public class Main {  public static void main(String[] args) {  String text = "маркиза де помпадур любила говорить: после нас хоть потоп.";  String[] words = text.split("\\s+");  StringBuilder result = new StringBuilder();  for (String word : words) {  char firstChar = word.charAt(0);  String modifiedWord = String.valueOf(firstChar);  for (int i = 1; i < word.length(); i++) {  char currentChar = word.charAt(i);  if (currentChar != firstChar) {  modifiedWord += currentChar;  }  }  result.append(modifiedWord).append(" ");  }  System.out.println(result.toString().trim());  }  } |

8

|  |
| --- |
| import java.util.List;  import java.util.ArrayList;  public class Main {  public static void main(String[] args) {  String text = "Исключить из текста подстроку максимальной длины, начинающуюся и заканчивающуюся одним и тем же символом";  // Find all substrings that start and end with the same character  List<String> substrings = new ArrayList<>();  for (int i = 0; i < text.length(); i++) {  char currentChar = text.charAt(i);  for (int j = i + 1; j < text.length(); j++) {  if (text.charAt(j) == currentChar) {  substrings.add(text.substring(i, j + 1));  }  }  }  // Find the longest substring  String longestSubstring = "";  for (String substring : substrings) {  if (substring.length() > longestSubstring.length()) {  longestSubstring = substring;  }  }  // Remove the longest substring from the original text  String result = text.replace(longestSubstring, "");  System.out.println(result);  }  } |