|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

**Институт информационных технологий (ИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиППО)**

**Дисциплина «Программирование на языке Джава»**

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5**

Выполнил студент группы ИНБО-02-20 Бойко К.В.

Принял Степанов П.В.

Практическая работа выполнена «\_\_\_»\_\_\_\_\_\_\_2021г.

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_» «\_\_\_»\_\_\_\_\_\_\_2021г.

Отметка о выполнении

**Москва – 2021 г.**

1. **Задание**

Упражнения по этой теме выполняются следующим образом: каждый учащийся выполняет от 3 до 5 задач, начиная с номера варианта задания, который соответствует номеру учащегося в журнале группы.

1. Треугольная последовательность

Дана монотонная последовательность, в которой каждое натуральное число k встречается ровно k раз: 1, 2, 2, 3, 3, 3, 4, 4, 4, 4,…

По данному натуральному n выведите первые n членов этой последовательности. Попробуйте обойтись только одним циклом for.

2. От 1 до n

Дано натуральное число n. Выведите все числа от 1 до n.

3. От A до B

Даны два целых числа A и В (каждое в отдельной строке). Выведите все числа от A до B включительно, в порядке возрастания, если A < B, или в порядке убывания в противном случае.

4. Заданная сумма цифр

Даны натуральные числа k и s. Определите, сколько существует k-значных натуральных чисел, сумма цифр которых равна d. Запись натурального числа не может начинаться с цифры 0.

В этой задаче можно использовать цикл для перебора всех цифр, стоящих на какой-либо позиции.

5. Сумма цифр числа

Дано натуральное число N. Вычислите сумму его цифр.

При решении этой задачи нельзя использовать строки, списки, массивы (ну и циклы, разумеется).

6. Проверка числа на простоту

Дано натуральное число n>1. Проверьте, является ли оно простым. Программа должна вывести слово YES, если число простое и NO, если число составное. Алгоритм должен иметь сложность O(logn).

Указание. Понятно, что задача сама по себе нерекурсивна, т.к. проверка числа n на простоту никак не сводится к проверке на простоту меньших чисел. Поэтому нужно сделать еще один параметр рекурсии: делитель числа, и именно по этому параметру и делать рекурсию.

7. Разложение на множители

Дано натуральное число n>1. Выведите все простые множители этого числа в порядке не убывания с учетом кратности. Алгоритм должен иметь сложность O(logn)

8. Палиндром

Дано слово, состоящее только из строчных латинских букв. Проверьте, является ли это слово палиндромом. Выведите YES или NO. При решении этой задачи нельзя пользоваться циклами, в решениях на питоне нельзя использовать срезы с шагом, отличным от 1.

1. **Ход Работы**

(основной код доступен по ссылке <https://github.com/Zyablikitsme/Java_practice>)

В ходе выполнения работы были получены следующие исходные коды:

**Recursion.java (**часть кода**)**

public static void rec1(int n) { *rec1*(n, 0); }  
  
public static void rec1(int n, int numbe){  
 for (int i = 1; i <= numbe; i++){  
 if (n > 0) n--;  
 else break;  
 System.*out*.print(numbe + " ");  
 }  
  
 if (n > 0) *rec1*(n, numbe+1);  
}  
  
public static void rec2(int n){  
 if (n != 1) *rec2*(n-1);  
 System.*out*.print(n + " ");  
}  
  
public static void rec3(int A, int B){  
 System.*out*.println(A);  
  
 if (A < B) *rec3*(++A, B);  
 else if (A > B) *rec3*(--A, B);  
}  
  
public static int rec4(int len, int sum, int k, int s) {  
 if (len == k) {  
 if (sum == s) {  
 return 1;  
 } else {  
 return 0;  
 }  
 }  
 int c = (len == 0 ? 1 : 0);  
 int res = 0;  
  
 for (int i = c; i < 10; i++) {  
 res += *rec4*(len + 1, sum + i, k, s);  
 }  
 return res;  
}  
  
public static int rec5(int n){  
 int sum = 0;  
 if (n > 0) {  
 sum += n % 10;  
 n /= 10;  
 }  
 if (n > 0) sum += *rec5*(n);  
 return sum;  
}  
  
public static String rec6(int n, int i) {  
 if (n < 2) return "NO";  
 else if (n == 2) return "YES";  
 else if (n % i == 0) return "NO";  
 else if (i < n/2) return *rec6*(n, i + 1);  
 else return "YES";  
}  
  
public static void rec7(int n, int k) {  
 if (k > n / 2) {  
 System.*out*.println(n);  
 return;  
 }  
 if (n % k == 0) {  
 System.*out*.println(k);  
 *rec7*(n / k, k);  
 }  
 else *rec7*(n, ++k);  
}  
  
public static String rec8(String word){ // OK  
 return *rec8*(word, 0);  
}  
  
public static String rec8(String word, int checkLetter){  
 if (word.charAt(checkLetter) != word.charAt(word.length() - checkLetter - 1)) return "NO";  
 if (checkLetter == word.length() / 2 - 1) return "YES";  
 return *rec8*(word, ++checkLetter);  
}

1. **Вывод**

В ходе данной работы мной были получены знания по разработке рекурсивных алгоритмов на Java.