Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Лабораторная работа № 5

по дисциплине: «Теоретико-числовые методы в криптографии»

на тему: «Символы Лежандра и Якоби»

Выполнил

обучающийся гр. ВКБ41

Якушевский Сергей Сергеевич

Проверила:

Ст. пр. Артамонова Е.А.

Лабораторная работа № 5

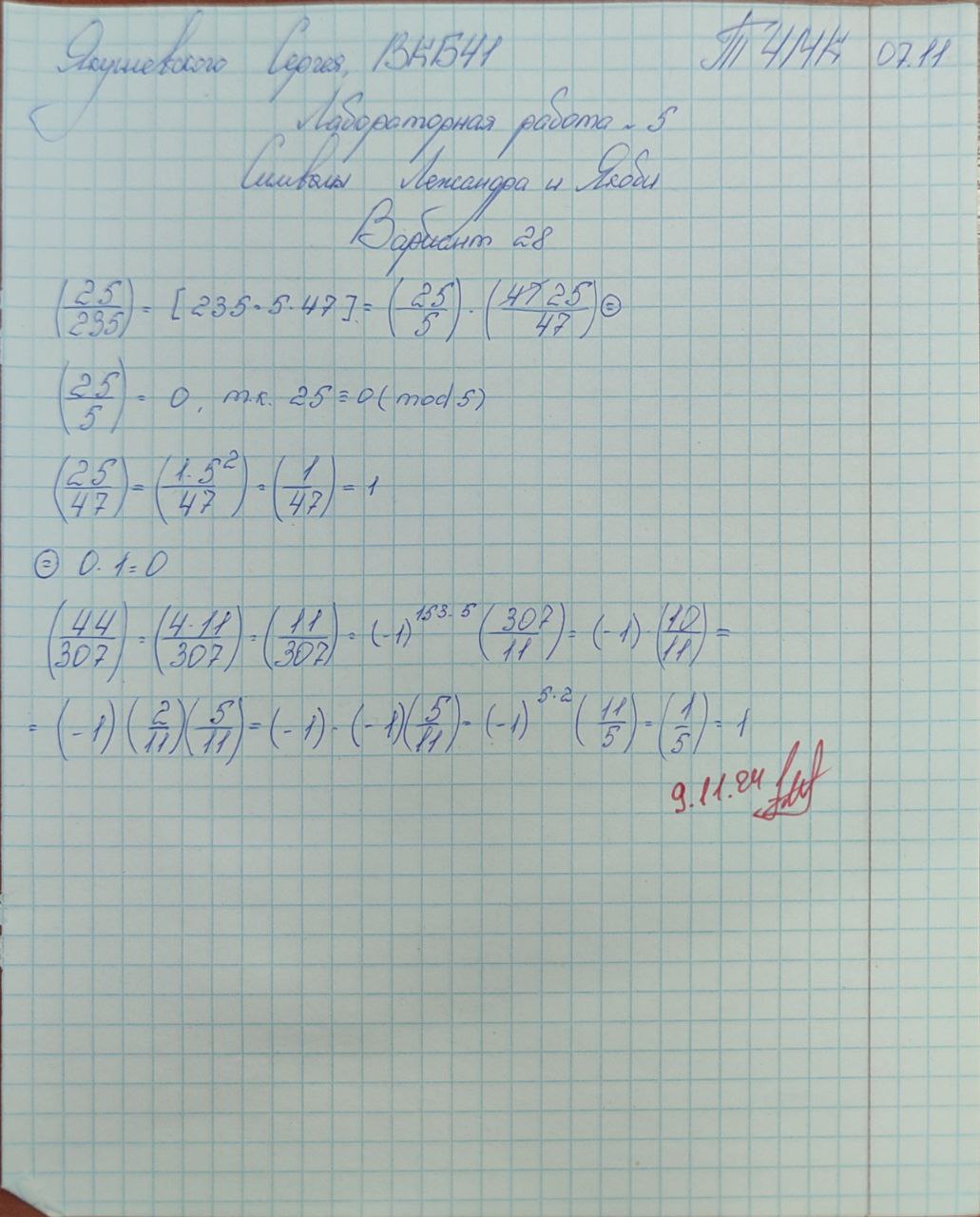
«Символы Лежандра и Якоби»

Вариант № 28

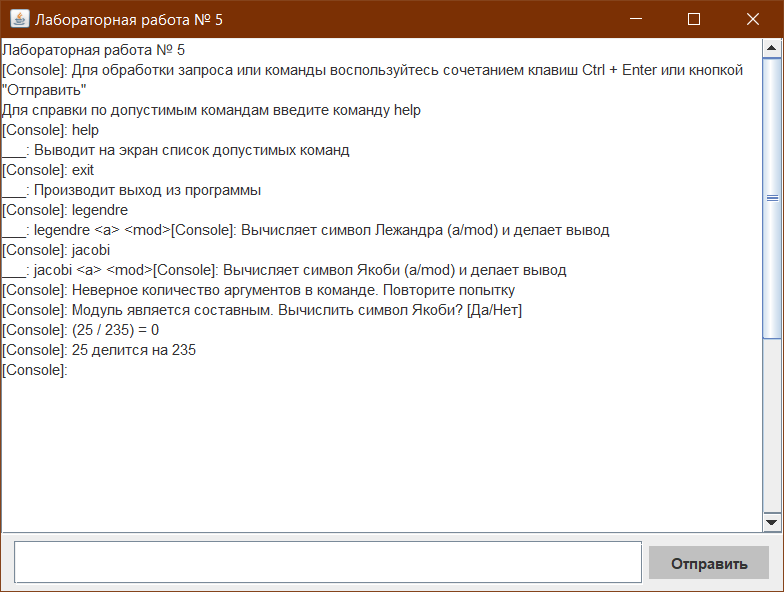
Задание.

Найти символы Лежандра и Якоби. Написать программу. Реализующую поиск символов Лежандра и Якоби.

## Ручная реализация.



## Входные данные и результат.



## Листинг программы.

package comandHandler;  
  
import exceptions.WrongArgumentsException;  
import mathUtils.MathUtils;  
import utils.Constants;  
import utils.ExceptionConstants;  
import utils.Printable;  
  
import java.util.ArrayList;  
import java.util.Arrays;  
  
public class CommandHandler implements Constants, ExceptionConstants {  
 private final Printable printable;  
 private CommandEnum lastCommand;  
 private String[] lastTokens;  
  
 public CommandHandler(Printable printable){  
 this.printable = printable;  
 }  
  
 public void execute(String request){  
 String[] tokens = request.split("\\s+");  
 try {  
 if(lastCommand == CommandEnum.*LEGENDRE* &&  
 tokens.length == 1 &&  
 tokens[0].equalsIgnoreCase("да")){  
 int a = Integer.*parseInt*(lastTokens[1]);  
 int mod = Integer.*parseInt*(lastTokens[2]);  
 printJacobi(a, mod);  
 lastCommand = null;  
 return;  
 }  
 if(lastCommand == CommandEnum.*JACOBI* &&  
 tokens.length == 1 &&  
 tokens[0].equalsIgnoreCase("да")){  
 int a = Integer.*parseInt*(lastTokens[1]);  
 int mod = Integer.*parseInt*(lastTokens[2]);  
 printLegendre(a, mod);  
 lastCommand = null;  
 return;  
 }  
 switch (CommandEnum.*valueOf*(tokens[0].toUpperCase())){  
 case *HELP* -> {  
 lastCommand = CommandEnum.*HELP*;  
 if(tokens.length == 1){  
 Arrays.*stream*(CommandEnum.*values*())  
 .map(CommandEnum::toString)  
 .forEach(printable::print);  
 return;  
 }  
 if(tokens.length == 2){  
 printable.print(CommandEnum.*valueOf*(tokens[1].toUpperCase()).toString());  
 };  
 }  
 case *LEGENDRE* -> {  
 if(tokens.length != 3){  
 throw new WrongArgumentsException(*WRONG\_ARGUMENTS\_AMOUNT\_MESSAGE*);  
 }  
 int mod = Integer.*parseInt*(tokens[2]);  
 if(!MathUtils.*isPrime*(mod)){  
 lastCommand = CommandEnum.*LEGENDRE*;  
 lastTokens = tokens;  
 printable.print("Модуль является составным. Вычислить символ Якоби? [Да/Нет]");  
 return;  
 }  
 int a = Integer.*parseInt*(tokens[1]);  
 printLegendre(a, mod);  
 }  
 case *JACOBI* -> {  
 if(tokens.length != 3){  
 throw new WrongArgumentsException(*WRONG\_ARGUMENTS\_AMOUNT\_MESSAGE*);  
 }  
 int mod = Integer.*parseInt*(tokens[2]);  
 if(MathUtils.*isPrime*(mod)){  
 lastCommand = CommandEnum.*JACOBI*;  
 lastTokens = tokens;  
 printable.print("Модуль является простым. Вычислить символ Лежандра? [Да/Нет]");  
 return;  
 }  
 int a = Integer.*parseInt*(tokens[1]);  
 printJacobi(a, mod);  
 }  
 case *EXIT* -> System.*exit*(0);  
 }  
 }catch (IllegalArgumentException ex){  
 lastCommand = null;  
 lastTokens = null;  
 printable.print(*WRONG\_COMMAND\_MESSAGE*);  
 }catch (ClassCastException ex){  
 printable.print(*INVALID\_ARGUMENTS\_MESSAGE*);  
 }catch (WrongArgumentsException ex){  
 printable.print(ex.getMessage());  
 }  
 }  
  
  
 private void printJacobi(int a, int mod){  
 int jac = MathUtils.*getJacobiSymbol*(a, mod);  
 printable.print(String.*format*("(%s / %s) = %s", a, mod, jac));  
 if(jac == 0){  
 printable.print(String.*format*("%s делится на %s", a, mod));  
 return;  
 }  
 if (jac == -1){  
 printable.print(String.*format*("%s является квадратичным невычетом по модулю %s", a, mod));  
 return;  
 }  
 printable.print(String.*format*("%s является квадратичным вычетом по модулю %s", a, mod));  
 }  
  
 private void printLegendre(int a, int mod){  
 int leg = MathUtils.*getLegendreSymbol*(a, mod);  
 printable.print(String.*format*("(%s / %s) = %s", a, mod, leg));  
 }  
  
}

## Вывод по работе.

В ходе выполнения лабораторной работы были закреплены теоретические сведения о способах выяснения наличия решений сравнения второй степени путём нахождения символов Лежандра и Якоби. Также, были получены практические навыки в программной реализации нахождения символов Лежандра и Якоби. Результатом лабораторной работы стал программный продукт, выполняющий поставленные в начале работы задачи.