Министерство цифрового развития связи и массовых коммуникаций РФ

Государственное бюджетное образовательное учреждение высшего образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Отчет по лабораторной работе №4
по дисциплине «Структуры и алгоритмы обработки данных»

Выполнил: студент группы БВТ1901

Кускова А. Е.

Руководитель:

Мелехин А. А.

Москва

2021

Задание:

Реализовать следующие структуры данных:

● Стек (stack): операции для стека: инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала;

● Дек (двусторонняя очередь, deque): операции для дека: инициализация, проверка на пустоту, добавление нового элемента в начало, добавление нового элемента в конец, извлечение элемента из начала, извлечение элемента из конца.

Разработать программу обработки данных, содержащихся в заранее подготовленном txt-файле, в соответствии с заданиями, применив указанную в задании структуру данных. Результат работы программы вывести на экран и сохранить в отдельном txt-файле.

Техническое задание:

1. Отсортировать строки файла, содержащие названия книг, в алфавитном порядке с использованием двух деков.

2. Дек содержит последовательность символов для шифровки сообщений. Дан текстовый файл, содержащий зашифрованное сообщение. Пользуясь деком, расшифровать текст. Известно, что при шифровке каждый символ сообщения заменялся следующим за ним в деке по часовой стрелке через один.

3. Даны три стержня и n дисков различного размера. Диски можно надевать на стержни, образуя из них башни. Перенести n дисков со стержня A на стержень C, сохранив их первоначальный порядок. При переносе дисков необходимо соблюдать следующие правила:

- на каждом шаге со стержня на стержень переносить только один диск;

- диск нельзя помещать на диск меньшего размера;

- для промежуточного хранения можно использовать стержень B.

Реализовать алгоритм, используя три стека вместо стержней A, B, C. Информация о дисках хранится в исходном файле.

4. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс круглых скобок в тексте, используя стек.

5. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс квадратных скобок в тексте, используя дек.

6. Дан файл из символов. Используя стек, за один просмотр файла напечатать сначала все цифры, затем все буквы, и, наконец, все остальные символы, сохраняя исходный порядок в каждой группе символов.

7. Дан файл из целых чисел. Используя дек, за один просмотр файла напечатать сначала все отрицательные числа, затем все положительные числа, сохраняя исходный порядок в каждой группе.

8. Дан текстовый файл. Используя стек, сформировать новый текстовый файл, содержащий строки исходного файла, записанные в обратном порядке: первая строка становится последней, вторая – предпоследней и т.д.

9. Дан текстовый файл. Используя стек, вычислить значение логического выражения, записанного в текстовом файле в следующей форме:

< ЛВ > ::= T | F | (N<ЛВ>) | (<ЛВ>A<ЛВ>) | (<ЛВ>X<ЛВ>) | (<ЛВ>O<ЛВ>),

где буквами обозначены логические константы и операции:

T – True, F – False, N – Not, A – And, X – Xor, O – Or.

10. Дан текстовый файл. В текстовом файле записана формула следующего вида:

<Формула> ::= <Цифра> | M(<Формула>,<Формула>) | N(Формула>,<Формула>)

< Цифра > ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

где буквами обозначены функции:

M – определение максимума, N – определение минимума.

Используя стек, вычислить значение заданного выражения.

11. Дан текстовый файл. Используя стек, проверить, является ли содержимое текстового файла правильной записью формулы вида:

< Формула > ::= < Терм > | < Терм > + < Формула > | < Терм > - < Формула >

< Терм > ::= < Имя > | (< Формула >)

< Имя > ::= x | y | z

Выполнение:

Реализация стека:

```
import java.util.ArrayList;
import java.util.Arrays;
public class Stack<E> {
    private ArrayList<E> stack;
    private int size;
    public Stack() {
        stack = new ArrayList<>();
        size = 0;
    }
    public boolean isEmpty() {
        return this.size == 0;
    }
    //добавить элемент в верхнюю часть стека
    public void push(E element) {
        if (element != null) {
            this.stack.add(element);
            this.size++;
        }
    }
    //возвратим верхний эл. без удаления
    public E peek() {
        if (size != 0) {
            return this.stack.get(this.size-1);
        }
        System.out.println("Stack is empty");
        return null;
```

```
    }
    //снимаем верхний элемент (с удалением)
    public E pop() {
        if (size != 0) {
            E element = this.stack.get(this.size-1);
            this.stack.remove(this.size-1);
            this.size--;
            return element;
        }
        System.out.println("Stack is empty");
        return null;
    }
    public String toString() {
        return Arrays.toString(this.stack.toArray());
    }
    public int getSize() {
        return this.size;
    }
}
```

Реализация дека:

```
import java.util.ArrayList;
import java.util.Arrays;

public class Deque<E> {
    private ArrayList<E> deque;
    private int size;
    public Deque() {
```

```
    this.deque = new ArrayList<>();
    this.size = 0;
}
public boolean isEmpty() {
    return this.size == 0;
}
//добавим последним
public void addLast(E element) {
    this.deque.add(element);
    this.size++;
}
//добавим первым
public void addFirst(E element) {
    this.deque.add(0, element);
    this.size++;
}
//вернем последний
public E getLast() {
    if(size != 0) {
        return this.deque.get(this.size-1);
    }
    System.out.println("Deque is empty");
    return null;
}
//вернем первый
public E getFirst() {
    if(size != 0) {
        return this.deque.get(0);
    }
```

```
    System.out.println("Deque is empty");
    return null;
}
//удалим последний
public E removeLast() {
    if(size != 0) {
        E element = this.deque.get(this.size-1);
        this.deque.remove(this.size-1);
        this.size--;
        return element;
    }
    System.out.println("Deque is empty");
    return null;
}
//удалим первый
public E removeFirst() {
    if(size != 0) {
        E element = this.deque.get(0);
        this.deque.remove(0);
        this.size--;
        return element;
    }
    System.out.println("Deque is empty");
    return null;
}
public String toString() {
    return Arrays.toString(this.deque.toArray());
}
public int getSize() {
```

```
        return this.size;
    }
}
```

Реализация тасков:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Scanner;
import java.util.Collections;
import java.io.*;

public class Lab4 {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        System.out.print("Введите номер таска: ");
        String task = input.nextLine();
        if(task.equals("1")) {
            task1(input);
        }
        if(task.equals("2")) {
            task2(input);
        }
        if(task.equals("3")) {
            task3(input);
        }
        if(task.equals("4")) {
            task4(input);
        }
```

```
    if(task.equals("5")) {
        task5(input);
    }
    if(task.equals("6")) {
        task6(input);
    }
    if(task.equals("7")) {
        task7(input);
    }
    if(task.equals("8")) {
        task8(input);
    }
    if(task.equals("9")) {
        task9(input);
    }
    if(task.equals("10")) {
        task10(input);
    }
    if(task.equals("11")) {
        task11(input);
    }
    input.close();
}

public static void task1(Scanner input) {
    BufferedReader reader;
    Deque<String> deq1 = new Deque<>();
    while(true) {
        try {
```

```
            System.out.print("Введите путь до файла: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь");
        }
    }
    try {
        String line = reader.readLine();
        while(line != null) {
            deq1.addLast(line);
            line = reader.readLine();

        }
    }
    catch(IOException ioExc) {
        ioExc.printStackTrace();
    }

    Deque<String> deq2 = new Deque<>();
    deq2.addFirst(deq1.removeFirst());

    while(!deq1.isEmpty()) {
        String first = deq1.getFirst().toLowerCase();
        String second = deq2.getFirst().toLowerCase();
        boolean compareWithLast = false;
```

```
if (deq1.getFirst().length() >= deq2.getFirst().length()) {
    for (int i = 0; i < second.length(); i++) {
        if (first.charAt(i) < second.charAt(i)) {
            deq2.addFirst(deq1.getFirst());
            deq1.removeFirst();
            break;
        }
        if (first.charAt(i) > second.charAt(i)) {
            compareWithLast = true;
            break;
        }
    }
}
else {
    for (int i = 0; i < first.length(); i++) {
        if (first.charAt(i) < second.charAt(i)) {
            deq2.addFirst(deq1.getFirst());
            deq1.removeFirst();
            break;
        }
        if (first.charAt(i) > second.charAt(i)) {
            compareWithLast = true;
            break;
        }
    }
}
if (compareWithLast) {
    second = deq2.getLast().toLowerCase();
}
```

```
        if (deq1.getFirst().length() >= deq2.getFirst().length() &&
compareWithLast) {
            for (int i = 0; i < second.length(); i++) {
                if (first.charAt(i) > second.charAt(i)) {
                    deq2.addLast(deq1.getFirst());
                    deq1.removeFirst();
                    break;
                }
                if (first.charAt(i) < second.charAt(i)) {
                    deq1.addLast(deq2.removeLast());
                    break;
                }
            }
        }
        else if (compareWithLast) {
            for(int i = 0; i < first.length(); i++) {
                if (first.charAt(i) > second.charAt(i)) {
                    deq2.addLast(deq1.getFirst());
                    deq1.removeFirst();
                    break;
                }
                if (first.charAt(i) < second.charAt(i)) {
                    deq1.addLast(deq2.removeLast());
                    break;
                }
            }
        }
    }
    System.out.println(deq2.toString());
```

```
}

public static void task2(Scanner input) {
    BufferedReader reader;
    Deque<Character> deq = new Deque<>();
    System.out.print("Введите строку: ");
    String decoder = input.nextLine().toLowerCase();
    for(int i = 0; i < decoder.length(); i++) {
        deq.addLast(decoder.charAt(i));
    }
    while(true) {
        try {
            System.out.print("Укажите путь до файла: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь до файла");
        }
    }
    String line = "";
    try {
        String newLine = reader.readLine();
        while (newLine != null) {
            line += newLine + " ";
            newLine = reader.readLine();
        }
```

```
        }
        catch (IOException ioExc) {
            ioExc.printStackTrace();
        }
        System.out.println("Закодированное сообщение:");
        System.out.println(line);
        String decodedMessage = "";
        line = line.toLowerCase();
        line = line.trim();
        boolean canDecode = true;
        int index = 0;
        while (decodedMessage.length() < line.length() && canDecode) {
            canDecode = false;
            if (line.charAt(index) == ' ') {
                index++;
                decodedMessage += " ";
            }
            for (int i = 0; i < deq.getSize(); i++) {
                if (deq.getFirst() == line.charAt(index)) {
                    canDecode = true;
                    break;
                }
                deq.addLast(deq.removeFirst());
            }
            if (!canDecode) {
                System.out.println("Не удается декодировать входное сообщение из-
за отсутствия символов в декодере");
                break;
            }
```

```
            deq.addLast(deq.removeFirst());
            deq.addLast(deq.removeFirst());
            decodedMessage += deq.getFirst();
            index++;
        }
        if (canDecode) {
            System.out.println("Расшифрованное сообщение:");
            System.out.println(decodedMessage);
        }
    }

    public static void task3 (Scanner input) {
        BufferedReader reader;
        ArrayList<Stack<Integer>> stacks = new ArrayList<Stack<Integer>>();
        stacks.add(new Stack<Integer>());
        stacks.add(new Stack<Integer>());
        stacks.add(new Stack<Integer>());
        while(true) {
            try {
                System.out.print("Укажите путь до файла: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch (IOException ioExc) {
                System.out.println("Неверный путь до файла");
            }
        }
```

```
try {
    ArrayList<Integer> disks = new ArrayList<>();
    String line = reader.readLine();
    String[] numbers;
    while (line != null) {
        line = line.trim();
        numbers = line.split(" ");
        for(int i = 0; i < numbers.length; i++) {
            disks.add(Integer.parseInt(numbers[i]));
        }
        line = reader.readLine();

    }
    Collections.sort(disks, Collections.reverseOrder());
    for (int i = 0; i < disks.size(); i++) {
        stacks.get(0).push(disks.get(i));
    }
}
catch(IOException ioExc) {
    ioExc.printStackTrace();
}
catch(NumberFormatException numExc) {
    System.out.println("Некорректный формат чисел в файле");
}
System.out.println(stacks.get(0).toString());
int count = stacks.get(0).getSize();
hanoiTowers(count, 0, 2, 1, stacks);
System.out.print(stacks.get(0).toString());
System.out.print(stacks.get(1).toString());
```

```
        System.out.println(stacks.get(2).toString());
    }


    public static void hanoiTowers(int count, int start, int middle, int end,
ArrayList<Stack<Integer>> stacks) {
        if(count > 0) {
            hanoiTowers(count-1, start, end, middle, stacks);
            stacks.get(middle).push(stacks.get(start).pop());
            hanoiTowers(count-1, end, middle , start, stacks);


        }
    }
    public static boolean task4(Scanner input) {
        BufferedReader reader;
        while(true) {
            try {
                System.out.print("Укажите путь до файла: ");
                String path = input.nextLine();
                input.close();
                reader = new BufferedReader(new FileReader(path));
                break;
            }
            catch(IOException ioExc)
            {
                System.out.println("Неверный путь до файла");
            }
        }
        String line = "";
        try {
```

```
    String newLine = reader.readLine();
    while(newLine != null) {
        line += newLine + "\n";
        newLine = reader.readLine();
    }
}
catch(IOException ioExc) {
    ioExc.printStackTrace();
}
System.out.println("Код программы:");
System.out.println(line);

Stack<Character> stack = new Stack<>();
for(int i = 0; i < line.length(); i++) {
    if(line.charAt(i) == '(') {
        stack.push('(');
    }
    if(line.charAt(i) == ')') {
        if(stack.getSize() != 0) {
            stack.pop();
        }
        else {
            System.out.println("Ожидается '('");
            return false;
        }
    }
}
if (stack.getSize() != 0) {
    System.out.println("Ожидается ')'");
```

```
        return false;
    }
    System.out.println("Код верен");
    return true;
}

public static boolean task5(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь до файла: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь до файла");
        }
    }
    String line = "";
    try {
        String newLine = reader.readLine();
        while(newLine != null) {
            line += newLine + "\n";
            newLine = reader.readLine();
        }
    }
    catch(IOException ioExc) {
```

```
            ioExc.printStackTrace();
        }
        System.out.println("Код программы:");
        System.out.println(line);

        Deque<Character> deq = new Deque<>();
        for(int i = 0; i < line.length(); i++) {
            if(line.charAt(i) == '[') {
                deq.addLast('[');
            }
            if(line.charAt(i) == ']') {
                if(deq.getSize() != 0) {
                    deq.removeLast();
                }
                else {
                    System.out.println("Ожидается '['");
                    return false;
                }
            }
        }
        if(deq.getSize() != 0) {
            System.out.println("Ожидается ']'");
            return false;
        }
        System.out.println("Код верен");
        return true;
    }
```

```
public static void task6(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь до файла: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь до файла");
        }
    }
    String line = "";
    try {
        String newLine = reader.readLine();
        while (newLine != null) {
            line += newLine;
            newLine = reader.readLine();
        }
    }
    catch (IOException ioExc) {
        ioExc.printStackTrace();
    }

    System.out.println("Исходный текст:");
    System.out.println(line);
```

```
        Stack<Character> stack = new Stack<>();
        for (int i = 0; i < line.length(); i++) {
            if (stack.peek() == null) {
                stack.push(line.charAt(i));
            }
            else {
                String storage = "";
                if(Character.isDigit(line.charAt(i))) {
                    while (stack.peek() != null && Character.isDigit(stack.peek())) {
                        storage += stack.pop();
                    }
                    stack.push(line.charAt(i));
                    for(int j = storage.length() - 1; j >= 0; j--) {
                        stack.push(storage.charAt(j));
                    }
                }
                if(Character.isLetter(line.charAt(i))) {
                    while(stack.peek() != null &&
Character.isLetterOrDigit(stack.peek())) {
                        storage += stack.pop();
                    }
                    stack.push(line.charAt(i));
                    for(int j = storage.length() - 1; j >= 0; j--) {
                        stack.push(storage.charAt(j));
                    }
                }
                if(!Character.isDigit(line.charAt(i)) &&
!Character.isLetter(line.charAt(i))) {
```

```
                while(stack.peek() != null) {
                    storage += stack.pop();
                }

                stack.push(line.charAt(i));

                for(int j = storage.length() - 1; j >= 0; j--) {
                    stack.push(storage.charAt(j));
                }
            }
        }
    }
    System.out.println("Новый порядок символов:");
    System.out.println(stack.toString());
}

public static void task7(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь к файлу: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь");
        }
```

```
        }
        String line = "";
        ArrayList<Integer> numbers = new ArrayList<>();
        try {
            String newLine = reader.readLine();
            while (newLine != null) {
                line += newLine + " ";
                newLine = reader.readLine();
            }
        }
        catch(IOException ioExc) {
            ioExc.printStackTrace();
        }
        String number = "";
        for(int i = 0; i < line.length(); i++) {
            if(line.charAt(i) == '-' && number.length() == 0 ||
Character.isDigit(line.charAt(i))) {
                number += line.charAt(i);
            }
            else {
                if(!number.equals("-") && number.length() != 0) {
                    numbers.add(Integer.parseInt(number));
                    number = "";
                }
            }
        }

        System.out.println("Исходный порядок чисел:");
        System.out.println(Arrays.toString(numbers.toArray()));
```

```
    Deque<Integer> deq = new Deque<>();
    for (int i = 0; i < numbers.size(); i++) {
        if (numbers.get(i) >= 0) {
            deq.addFirst(numbers.get(i));
        }
        else {
            deq.addLast(numbers.get(i));
        }
    }
    while (deq.getFirst() >= 0) {
        deq.addLast(deq.removeFirst());
    }
    while (deq.getSize() != 0) {
        if(deq.getFirst() < 0) {
            System.out.print(deq.removeFirst());
            System.out.print(" ");
        }
        if (deq.getFirst() >= 0) {
            System.out.print(deq.removeLast());
            System.out.print(" ");
        }
    }
}

public static void task8(Scanner input) {
    Stack<String> stack = new Stack<>();
    BufferedReader reader;
    FileWriter writer;
    while(true) {
```

```
    try {
        System.out.print("Укажите путь до файла: ");
        String path = input.nextLine();
        reader = new BufferedReader(new FileReader(path));

        System.out.print("Укажите путь к выходному файлу: ");
        path = input.nextLine();
        input.close();
        writer = new FileWriter(path, false);
        break;
    }
    catch(IOException ioExc)
    {
        System.out.println("Неверный путь до файла");
    }
}
String line = "";
try {
    String newLine = reader.readLine();
    while(newLine != null) {
        stack.push(newLine);
        newLine = reader.readLine();
    }
    while(stack.peek() != null) {
        writer.write(stack.pop());
        writer.append('\n');
    }
    writer.flush();
}
```

```
    catch(IOException ioExc) {
        ioExc.printStackTrace();
    }
}

public static boolean task9(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь до файла: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь до файла");
        }
    }
    String line = "";
    try {
        line = reader.readLine();
    }
    catch (IOException ioExc) {
        ioExc.printStackTrace();
    }
    Stack<Character> stack = new Stack<>();

    for(int i = 0; i < line.length();) {
```

```
if (line.charAt(i) != ')') {
    if(line.charAt(i) != '(') {
        stack.push(line.charAt(i));
    }
    i++;
}

else if (stack.getSize() != 0) {
    char elem = stack.pop();
    char var = stack.peek();
    stack.push(elem);
    switch(var) {
        case 'N': {
            if(stack.peek() == 'T') {
                i++;
                stack.pop();
                stack.pop();
                stack.push('F');
                break;
            }
            else {
                i++;
                stack.pop();
                stack.pop();
                stack.push('T');
                break;
            }
        }
        case 'A': {
```

```
    if(stack.peek() == 'T') {
        stack.pop();
        stack.pop();
        if(stack.peek() == 'T') {
            i++;
            stack.pop();
            stack.push('T');
            break;
        }
        else {
            i++;
            stack.pop();
            stack.push('F');
            break;
        }
    }
    else {
        stack.pop();
        stack.pop();
        i++;
        stack.pop();
        stack.push('F');
        break;
    }
}
case 'X': {
    char first = stack.peek();
    stack.pop();
    stack.pop();
```

```
    char second = stack.peek();
    if (first == second) {
        i++;
        stack.pop();
        stack.push('F');
        break;
    } else {
        i++;
        stack.pop();
        stack.push('T');
        break;
    }
}
case 'O': {
    char first = stack.peek();
    stack.pop();
    stack.pop();
    char second = stack.peek();
    if (first == 'F' && second == 'F') {
        i++;
        stack.pop();
        stack.push('F');
        break;
    } else {
        i++;
        stack.pop();
        stack.push('T');
        break;
    }
```

```
            }
        }
    }
}

if(stack.peek() == 'T') {
    System.out.println("True");
    return true;
}
System.out.println("False");
return false;
}

public static int task10(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь до файла: ");
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch(IOException ioExc) {
            System.out.println("Неверный путь до файла");
        }
    }
    String line = "";
    try {
```

```
    line = reader.readLine();
}
catch(IOException ioExc) {
    ioExc.printStackTrace();
}

Stack<Character> stack = new Stack<>();
for(int i = 0; i < line.length();) {
    if (line.charAt(i) != ')') {
        if(line.charAt(i) != '(') {
            stack.push(line.charAt(i));
        }
        i++;
    }

    else if (stack.getSize() != 0) {
        char elem1 = stack.pop();
        stack.pop();
        char elem2 = stack.pop();
        char var = stack.pop();
        switch(var) {
            case 'N': {
                if (elem1 > elem2) {
                    i++;
                    stack.push(elem2);
                    break;
                } else {
                    i++;
                    stack.push(elem1);
```

```
                    break;
                }
            }
            case 'M': {
                if (elem1 > elem2) {
                    i++;
                    stack.push(elem1);
                    break;
                } else {
                    i++;
                    stack.push(elem2);
                    break;
                }
            }
        }
    }
    if (Character.isDigit(stack.peek())) {
        System.out.println(stack.peek());
        return stack.peek();
    }
    return 0;

}
public static boolean task11(Scanner input) {
    BufferedReader reader;
    while(true) {
        try {
            System.out.print("Укажите путь к файлу: ");
```

```
            String path = input.nextLine();
            input.close();
            reader = new BufferedReader(new FileReader(path));
            break;
        }
        catch (IOException ioExc) {
            System.out.println("Указан неверный путь");
        }
    }
    String line = "";
    try {
        line = reader.readLine();
    }
    catch(IOException ioExc) {
        ioExc.printStackTrace();
    }

    Stack<Character> stack = new Stack<>();
    for (int i = 0; i < line.length();) {
        if (line.charAt(i) != ')') {
            if (line.charAt(i) != '(') {
                stack.push(line.charAt(i));
            }
            i++;
        }

        else if(stack.getSize() != 0) {
            Character elem1 = stack.pop();
            Character var = stack.pop();
```

```
            Character elem2 = stack.peek();
            if(var == null || elem2 == null) {
               break;
            }
            if((elem1 != 'x' && elem1 != 'y' && elem1 != 'z') || (elem2 != 'x' &&
elem2 != 'y' && elem2 != 'z')) {
               break;
            }
            stack.push(var);
            stack.push(elem1);
            if(var == '+' || var == '-') {
               i++;
               stack.pop();
               stack.pop();
               stack.pop();
               stack.push('x');
            }
         }
      }
      if(stack.getSize() == 1 && (stack.peek() == 'x' || stack.peek() == 'y' ||
stack.peek() == 'z')) {
         System.out.println("True");
         return true;
      }
      System.out.println("False");
      return false;
   }
}
```

Вывод:

Разработали такие структуры данных, как стек и дек. Разработали программу обработки данных, содержащихся в заранее подготовленном txt-файле, в соответствии с заданиями, применив указанную в задании структуру данных.