Министерство цифрового развития, связи и массовых коммуникаций Государственное образовательного учреждение высшего образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Лабораторная работа № 4

«Реализация стека/дека»

Выполнила студентка

группы БВТ1903:

Симонова Катя

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.

Оформить отчет о лабораторной работе в ipynb или pdf-файле. **Задания:**

* 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 *А* на стержень *С***,** 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 *В***.**Реализовать алгоритм, используя три ***стека*** вместо стержней *А***,** *В***,** *С***.** Информация о дисках хранится в исходном файле.

* 1. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс круглых скобок в тексте, используя ***стек*.**
  2. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс квадратных скобок в тексте, используя ***дек*.**

1. 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.
2. 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.
3. 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.
4. 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.

1. 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

Код программы:

package com.company;  
  
  
import javax.script.ScriptException;  
import java.util.Deque;  
import java.util.LinkedList;  
  
public class Main {  
  
 public static void main(String[] args) throws ScriptException {  
 //task1  
 String str1="A C B D D D SD ";  
 new task1(str1);  
 System.out.println();  
  
 //task2  
 String str = "12315";  
 Deque<Character> dec = new LinkedList<>();  
 String code = ")а12ко39ш(5";  
 for (int i = 0; i < code.length(); i++) {  
 dec.addLast(code.charAt(i));  
 }  
 new task2(dec, str);  
  
 //task3  
 System.out.print("task 3: \n");  
 new task3(4, 'A','B','C');  
 //task4  
 String str4 = "((((1151()))()))";  
 new task4(str4);  
 //task 5  
 String str5 = "[]";  
 new task5(str5);  
 //task6  
 String str6 = "111qqqwww111--1q-\_+";  
 new task6(str6);  
 System.*out*.println();  
 //task7  
 int[] arr7 = {1, -15, 16, 71, -3};  
 new task7(arr7);  
 System.*out*.println();  
 //task8  
 String str8 = "15 q1 1521 qwt";  
 new task8(str8);  
 System.*out*.println();  
 //task9  
 String str9= "FO(TAFO(FOT))X(NT)"; // "T"  
 //new task9(str9);  
 task9.*mtask9*(str9);  
 //task10  
 String str10="N(9,(M(3,N(1,2))))"; // 3  
 task10.mtask10(str10);  
 //task11  
 String str11="x-((y+z)+(z-y))"; // 1-((2+3)+(3-2)) = -5  
 task11.mtask11(str11);  
 String str11b="x-((+z)+(z-y))";   
 task11.mtask11(str11b);  
 }  
}

package com.company;  
  
import java.util.Deque;  
import java.util.LinkedList;  
  
public class task1 {  
 public static boolean lt(String str1, String str2) {  
 return str1.compareTo(str2) < 0;  
 }  
  
 public task1(String Str) {  
 String[] words = Str.split(" ");  
 Deque<String> a = new LinkedList<>();  
 Deque<String> b = new LinkedList<>();  
 for (int i = 0; i < words.length; i++) {  
 a.push(words[i]);  
 }  
 String str = "";  
 while (!a.isEmpty()) {  
 for (int i = 0; i < a.size(); i++) {  
 if (*lt*(str, a.peekFirst())) {  
 str = a.peekFirst();  
 }  
 a.addLast(a.removeFirst());  
 }  
 for (int i = 0; i < a.size(); i++) {  
 if (str.equals(a.peekFirst())) {  
 a.removeFirst();  
 break;  
 }  
 a.addLast(a.removeFirst());  
 }  
 b.addFirst(str);  
 str="";  
 }  
 System.*out*.print("task 1: ");  
 while (!b.isEmpty()){  
 System.*out*.print(b.removeFirst()+" ");  
 }  
 }  
}

package com.company;  
  
import java.util.Deque;  
  
public class task2 {  
  
 public task2(Deque<Character> dec, String str) {  
 String str1 = "";  
 for (int i = 0; i < str.length(); i++) {  
 while (str1.length() < i + 1) {  
 if (str.charAt(i) == dec.peekLast()) {  
 dec.add(dec.pop());  
 dec.add(dec.pop());  
  
 str1 += dec.peekLast();  
 }  
 dec.add(dec.pop());  
 }  
 }  
 System.*out*.println("task 2: " + str1);  
 }  
  
}

package com.company;  
  
import java.util.Deque;  
import java.util.Stack;  
  
public class task3 {  
 public task3(int topN, char from, char inter, char to) {  
 if (topN == 1) {  
 System.*out*.println(" Диск 1 из " + from + " на " + to);  
 } else {  
 new task3(topN - 1, from, to, inter);  
 System.*out*.println(" Диск " + topN + " из " + from + " на " + to);  
 new task3(topN - 1, inter, from, to);  
 }  
 }  
}

package com.company;  
  
import java.util.Deque;  
import java.util.Stack;  
  
public class task4 {  
 public task4(String str) {  
 Stack stc = new Stack();  
 Stack stc1 = new Stack();  
 for(int i=0; i < str.length(); i++){  
 if(str.charAt(i) == '(')  
 stc.push(str.charAt(i));  
 else if(str.charAt(i) == ')')  
 stc1.push(str.charAt(i));  
 }  
 if(stc.size() == stc1.size())  
 System.*out*.println("task 4: true");  
 else  
 System.*out*.println("task 4: false");  
 }  
}

package com.company;  
  
import java.util.Deque;  
import java.util.LinkedList;  
import java.util.Stack;  
  
public class task5 {  
 public task5 (String str) {  
 Deque stc = new LinkedList();  
 Deque stc1 = new LinkedList();  
 for(int i=0; i < str.length(); i++){  
 if(str.charAt(i)=='[')  
 stc.push(str.charAt(i));  
 else if(str.charAt(i)==']')  
 stc1.push(str.charAt(i));  
 }  
 if(stc.size()==stc1.size())  
 System.*out*.println("task 5: true");  
 else  
 System.*out*.println("task 5: false");  
 }  
}

package com.company;  
  
import java.util.Stack;  
  
public class task6 {  
 public task6(String str) {  
 Stack num = new Stack();  
 Stack sim = new Stack();  
 Stack buk = new Stack();  
 Stack per = new Stack();  
 for (int i = 0; i < str.length(); i++) {  
 if (str.charAt(i) < '9' && '0' < str.charAt(i))  
 num.push(str.charAt(i));  
 else if ((str.charAt(i) < 'Z' && 'A' < str.charAt(i)) || (str.charAt(i) < 'z' && 'a' < str.charAt(i)))  
 buk.push(str.charAt(i));  
 else  
 sim.push(str.charAt(i));  
 }  
 System.*out*.print("task 6: ");  
 while (!num.empty()) {  
 per.push(num.pop());  
 }  
 while (!per.empty()) {  
 System.*out*.print(per.pop());  
 }while (!buk.empty()) {  
 per.push(buk.pop());  
 }  
 while (!per.empty()) {  
 System.*out*.print(per.pop());  
 }  
 while (!sim.empty()) {  
 per.push(sim.pop());  
 }  
 while (!per.empty()) {  
 System.*out*.print(per.pop());  
 }  
  
 }  
}

package com.company;  
  
import java.lang.reflect.Array;  
import java.util.Deque;  
import java.util.LinkedList;  
import java.util.Stack;  
  
public class task7 {  
 public task7(int[] arr) {  
 Deque<Integer> pos = new LinkedList<>();  
 Deque<Integer> neg = new LinkedList<>();  
 for (int i=0; i<arr.length; i++){  
 if(arr[i]<0)  
 neg.addLast(arr[i]);  
 else  
 pos.addFirst(arr[i]);  
 }  
 System.*out*.print("task 7: ");  
 for (int x : neg){  
 System.*out*.print(" " + x);  
 }  
 for (int x : pos){  
 System.*out*.print(" " + x);  
 }  
 }  
}

package com.company;  
  
import java.util.Stack;  
  
public class task8 {  
 public task8(String str) {  
 String[] words = str.split(" ");  
 Stack stc = new Stack();  
 System.*out*.print("task 8: ");  
 for (int i = words.length - 1; i >= 0; i--) {  
 System.*out*.print(stc.push(words[i]) + " ");  
 }  
 }  
}

package com.company;  
  
import java.util.Stack;  
  
public class task9 {  
  
 public static void compute(String expr, Stack<Boolean> st, int deep) {  
 int idx;  
 // String d = " ".repeat(deep);  
 // System.out.println(d + "EXPR: " + expr);  
 if (expr.equals("T")) {  
 st.push(true);  
 } else if (expr.equals("F")) {  
 st.push(false);  
 } else if (*isInBrackets*(expr)) {  
 *compute*(expr.substring(1, expr.length() - 1), st, deep + 1);  
 } else if (expr.startsWith("N")) {  
 *compute*(expr.substring(1), st, deep + 1);  
 st.push(!st.pop());  
 } else if ((idx = *indexWithBrackets*(expr,'O')) != -1) {  
 *compute*(expr.substring(0, idx), st, deep + 1);  
 *compute*(expr.substring(idx + 1), st, deep + 1);  
 st.push(st.pop() || st.pop());  
 } else if ((idx = *indexWithBrackets*(expr, 'X')) != -1) {  
 *compute*(expr.substring(0, idx), st, deep + 1);  
 *compute*(expr.substring(idx + 1), st, deep + 1);  
 st.push(st.pop() ^ st.pop());  
 } else if ((idx = *indexWithBrackets*(expr, 'A')) != -1) {  
 *compute*(expr.substring(0, idx), st, deep + 1);  
 *compute*(expr.substring(idx + 1), st, deep + 1);  
 st.push(st.pop() && st.pop());  
 } else {  
 throw new Error("Illegar expression");  
 }  
 }  
  
 public static int indexWithBrackets(String s, char ch) {  
 int bracket = 0;  
 for (int i = 0; i < s.length(); i++) {  
 if (s.charAt(i) == '(') { bracket += 1;}  
 else if (s.charAt(i) == ')') { bracket -= 1; }  
 else if (bracket == 0) { // not in brackets  
 if (s.charAt(i) == ch) {  
 return i;  
 }  
 }  
 }  
 return -1;  
 }  
  
 public static boolean isInBrackets(String s) {  
 if (!(s.startsWith("(") && s.endsWith(")"))) {  
 return false;  
 }  
 s = s.substring(1);  
 int bracket = 1;  
 for (int i = 0; i < s.length(); i++) {  
 if (bracket == 0) { return false; }  
 if (s.charAt(i) == '(') { bracket += 1;}  
 else if (s.charAt(i) == ')') { bracket -= 1; }  
  
 if (bracket < 0) { throw new Error("Illegar expression"); }  
 }  
 return bracket == 0;  
 }  
  
 public static void mtask9(String expr) {  
 Stack<Boolean> st = new Stack<>();  
 *compute*(expr, st, 0);  
 Boolean result = st.pop();  
 if (result == null) {  
 throw new Error("Illegar expression");  
 } else {  
 System.*out*.println("task 9: " + result);  
 }  
 }  
}

package com.company;  
  
import java.util.Stack;  
  
public class task10 {  
  
 public static void compute(String expr, Stack<Integer> st, int deep) {  
 int idx;  
 // String d = " ".repeat(deep);  
 // System.out.println(d + "EXPR: " + expr);  
 int num = -1;  
 boolean isNumber = true;  
 try {  
 num = Integer.*parseInt*(expr);  
 } catch (Exception e) {  
 isNumber = false;  
 }  
  
 if (isNumber) {  
 st.push(num);  
 } else if (*isInBrackets*(expr)) {  
 *compute*(expr.substring(1, expr.length() - 1), st, deep + 1);  
 } else if (expr.startsWith("M(") && expr.endsWith(")")) {  
 *compute*(expr.substring(2, expr.length() - 1), st, deep + 1);  
 st.push(Math.*max*(st.pop(), st.pop()));  
 } else if (expr.startsWith("N(") && expr.endsWith(")")) {  
 *compute*(expr.substring(2, expr.length() - 1), st, deep + 1);  
 st.push(Math.*min*(st.pop(), st.pop()));  
 } else if ((idx = *indexWithBrackets*(expr,',')) != -1) {  
 *compute*(expr.substring(0, idx), st, deep + 1);  
 *compute*(expr.substring(idx + 1), st, deep + 1);  
 } else {  
 throw new Error("Illegar expression");  
 }  
 }  
  
 public static int indexWithBrackets(String s, char ch) {  
 int bracket = 0;  
 for (int i = 0; i < s.length(); i++) {  
 if (s.charAt(i) == '(') { bracket += 1;}  
 else if (s.charAt(i) == ')') { bracket -= 1; }  
 else if (bracket == 0) { // not in brackets  
 if (s.charAt(i) == ch) {  
 return i;  
 }  
 }  
 }  
 return -1;  
 }  
  
 public static boolean isInBrackets(String s) {  
 if (!(s.startsWith("(") && s.endsWith(")"))) {  
 return false;  
 }  
 s = s.substring(1);  
 int bracket = 1;  
 for (int i = 0; i < s.length(); i++) {  
 if (bracket == 0) { return false; }  
 if (s.charAt(i) == '(') { bracket += 1;}  
 else if (s.charAt(i) == ')') { bracket -= 1; }  
  
 if (bracket < 0) { throw new Error("Illegar expression"); }  
 }  
 return bracket == 0;  
 }  
  
 public static void mtask10(String expr) {  
 Stack<Integer> st = new Stack<>();  
 *compute*(expr, st, 0);  
 int result = st.pop();  
 System.*out*.println("task 10: " + result);  
 }  
}

package com.company;  
  
import javax.script.ScriptEngine;  
import javax.script.ScriptEngineManager;  
import javax.script.ScriptException;  
import java.util.Stack;  
  
public class task11 {  
 public static void compute(String expr, Stack<Integer> st, int deep) throws Exception {  
// String d = " ".repeat(deep);  
// System.out.println(d + "EXPR: " + expr);  
 if (expr.equals("x")) {  
 st.push(1);  
 } else if (expr.equals("y")) {  
 st.push(2);  
 } else if (expr.equals("z")) {  
 st.push(3);  
 } else if (*isInBrackets*(expr)) {  
 *compute*(expr.substring(1, expr.length() - 1), st, deep + 1);  
 } else if ((*indexWithBrackets*(expr, '+') != -1) || (*indexWithBrackets*(expr, '-') != -1)) {  
 int idx = Math.*min*(  
 *maxIfM1*(*indexWithBrackets*(expr, '+')),  
 *maxIfM1*(*indexWithBrackets*(expr, '-'))  
 );  
 *compute*(expr.substring(0, idx), st, deep + 1);  
 *compute*(expr.substring(idx + 1), st, deep + 1);  
 if (expr.charAt(idx) == '+') {  
 st.push(st.pop() + st.pop());  
 } else {  
 st.push(- st.pop() + st.pop());  
 }  
  
 } else {  
 throw new Exception("Illegar expression");  
 }  
 }  
  
 public static int maxIfM1(int x) {  
 if (x == -1) { return Integer.*MAX\_VALUE*; }  
 else { return x; }  
 }  
  
 public static int indexWithBrackets(String s, char ch) {  
 int bracket = 0;  
 for (int i = 0; i < s.length(); i++) {  
 if (s.charAt(i) == '(') { bracket += 1;}  
 else if (s.charAt(i) == ')') { bracket -= 1; }  
 else if (bracket == 0) { // not in brackets  
 if (s.charAt(i) == ch) {  
 return i;  
 }  
 }  
 }  
 return -1;  
 }  
  
 public static boolean isInBrackets(String s) throws Exception{  
 if (!(s.startsWith("(") && s.endsWith(")"))) {  
 return false;  
 }  
 s = s.substring(1);  
 int bracket = 1;  
 for (int i = 0; i < s.length(); i++) {  
 if (bracket == 0) { return false; }  
 if (s.charAt(i) == '(') { bracket += 1;}  
 else if (s.charAt(i) == ')') { bracket -= 1; }  
 if (bracket < 0) { throw new Exception("Illegal expression"); }  
 }  
 return bracket == 0;  
 }  
  
 public static void mtask11(String expr) {  
 Stack<Integer> st = new Stack<>();  
 try {  
 *compute*(expr, st, 0);  
 System.*out*.println("Task 11: Correct expression. When x = 1; y = 2; z = 3,\n\texpr = " + st.pop());  
 //System.out.println("Task 11: Correct expression.");  
 } catch (Exception e) {  
 System.*out*.println("Task 11: Illegal expression.");  
 }  
 }  
}

Снимки экрана выполнения программы

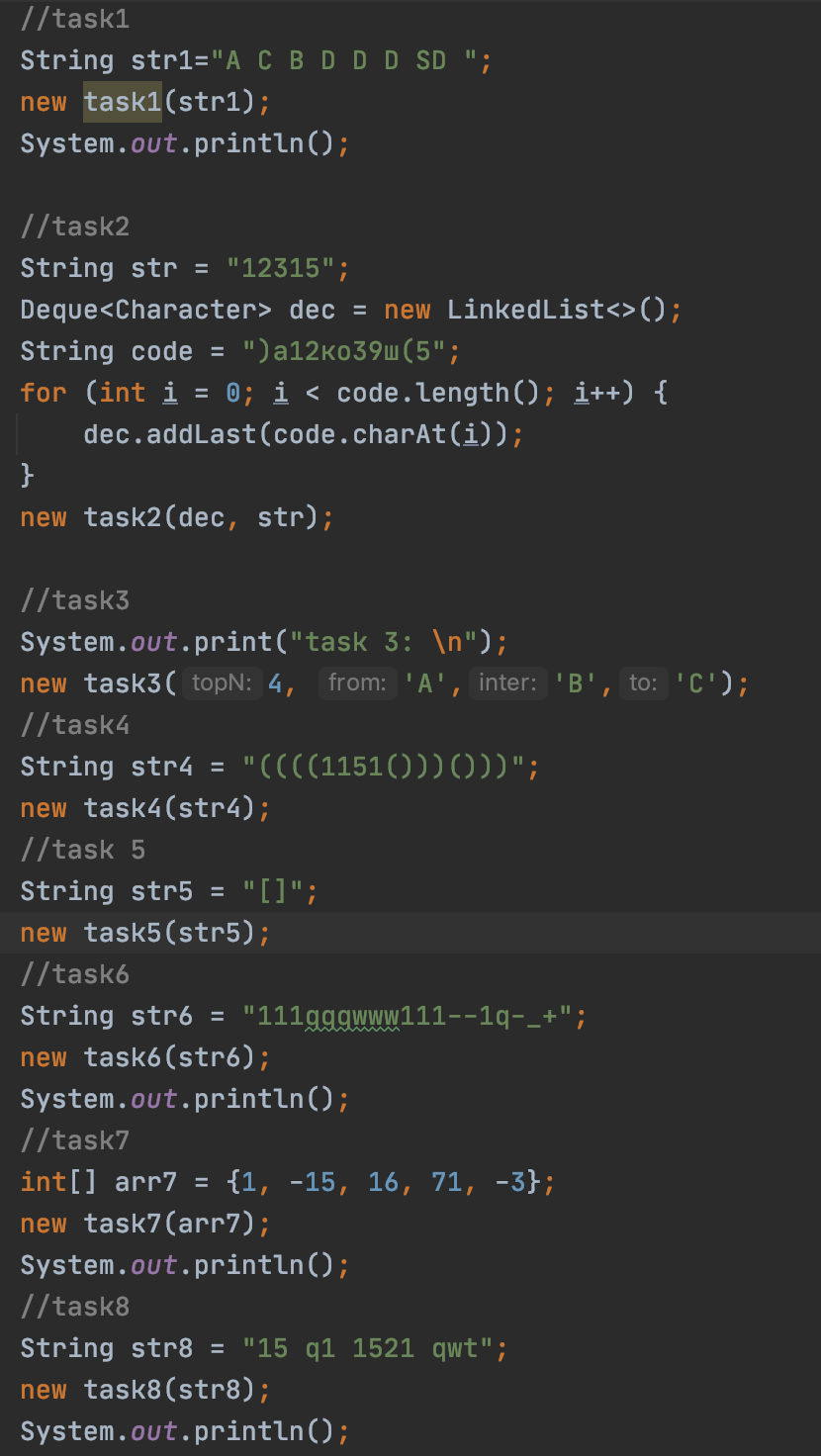
 

Рис. 1 Входные данные

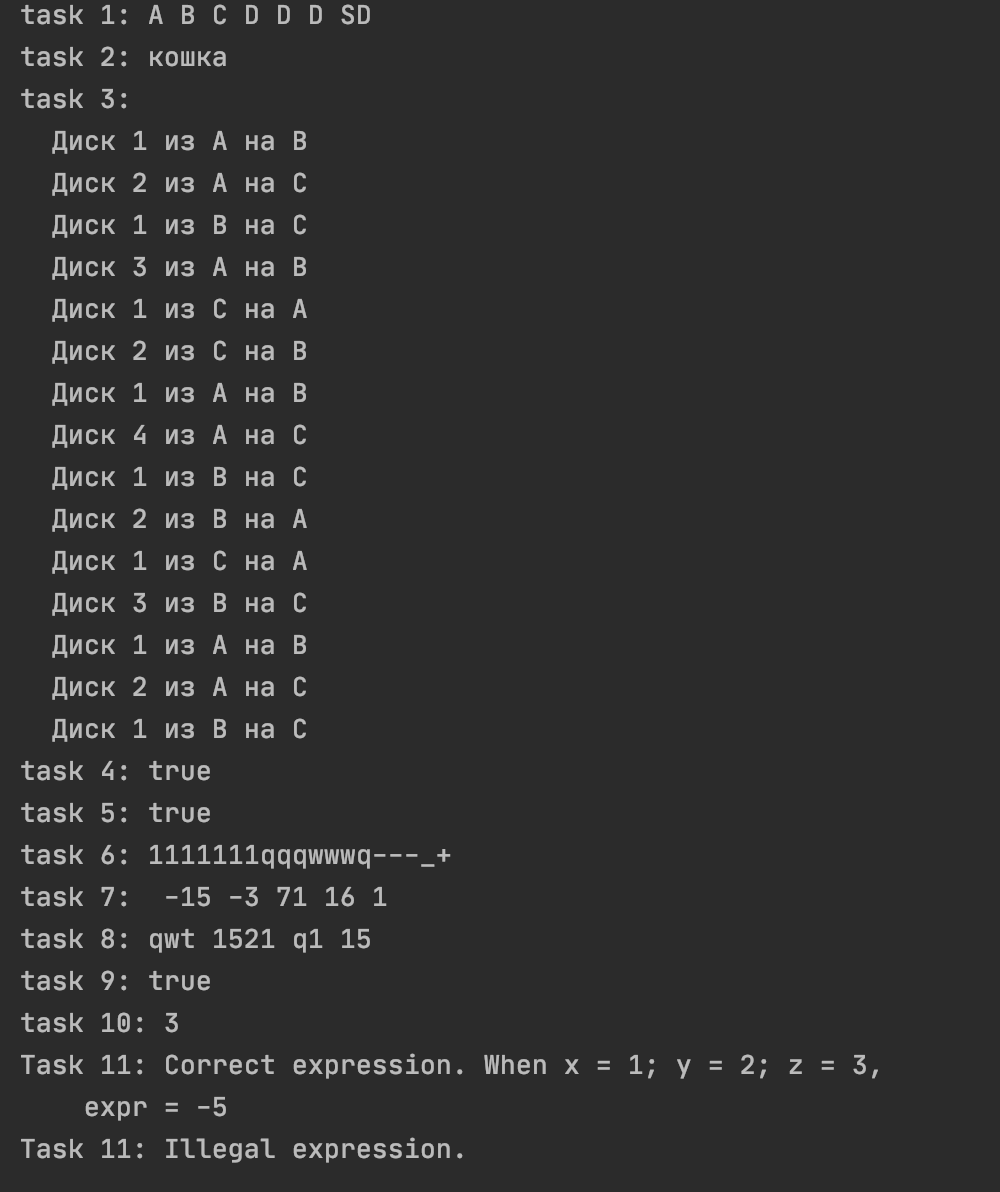


Рис. 2 Выходные данные

# Вывод

В ходе выполнения лабораторной работы я реализовала структуры стек и дек, при помощи них отсортировала строку в алфавитном порядке, расшифровала сообщение, решила задачу на диски и стержни, проверила соответствие скобок, рассортировала буквы, цифры, и отрицательные, и положительные, решила логическое выражение, нашла максимум и минимум и проверила выражение.