**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена Трудового Красного Знамени**

**федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра “Математическая кибернетика и информационные технологии”

Курсовая работа

по дисциплине: «Структуры и алгоритмы обработки данных»

Выполнил: студент группы БФИ2001

Марков А.К.

Проверил: Симонов С.Е.

**Вариант №14**

Москва 2022

**Цель работы**

Решить 10 задач в соответствии с вариантом.

**Ход работы**

# Задача 1

**Условие задания**

В очереди в аэропорту стоят n моржей. Они пронумерованы начиная с конца очереди: морж с номером 1 стоит в конце очереди, а морж с номером n — в начале. Морж с номером i имеет возраст ai.

Морж с номером i становится недовольным, если впереди него стоит более молодой морж, то есть существует такое j (i < j), что ai > aj. Недовольство i-го моржа равно количеству моржей между ним и самым удаленным моржом впереди, который моложе i-го. То есть чем дальше от него стоит этот молодой морж, тем сильнее недовольство.

Менеджер аэропорта попросил вас посчитать для каждого из n моржей в очереди его недовольство.

**Входные данные** В первой строке записано целое число n (2 ≤ n ≤ 105) — количество моржей в очереди. Во второй строке записаны целые числа ai (1 ≤ ai ≤ 109).

Обратите внимание, что некоторые моржи могут иметь одинаковый возраст, но для возникновения недовольства необходимо, чтобы тот морж, что ближе к началу, был строго моложе другого.

**Выходные данные** Выведите n чисел: если i-ый морж всем доволен, выведите «-1» (без кавычек). Иначе выведите недовольство i-го моржа: количество моржей, которые стоят между ним и самым удаленным моржом впереди него, который более молод.

**Примеры**

**входные данные**

6

10 8 5 3 50 45

**выходные данные**

2 1 0 -1 0 -1

**Листинг программы**

**#1**

**n** = **int(input())**

**line** = **list(map(int, input()**.**split()))**

def **task1(n, line):**

**rage** = **list(**-**i** for **i** in **line)**

**rage[n**-**1]**=-**1**

for **i** in **range(n):**

for **j** in **range(n**-**1, i,** -**1):**

if**(i**!=**j** and **line[j]**<**line[i]** and **rage[i]**<**0** and **i**!=**n**-**1):**

**rage[i]** = **j**-**i**-**1**

break

else**:**

**rage[i]** =-**1**

**#print(rage)**

for **k** in **range(len(rage)):**

**print(rage[k], " ", end**=**'')**

**task1(n, line)**

# Задача 5

**Условие задания**

Текстовый редактор Beroffice имеет широкие возможности по работе с текстами. Одна из возможностей — автоматический поиск опечаток и формирование предложения по их исправлению.

При наборе текста в Beroffice используются только строчные буквы английского алфавита (то есть 26 букв от a до z). При наборе слова Beroffice предполагает, что слово набрано с опечаткой, если встречаются три или более согласные буквы подряд. Единственное исключение — если блок подряд идущих согласных букв состоит из одинаковых букв, то этот блок (даже если его длина больше или равна трём) не считается опечаткой. Формально, слово набрано с опечаткой, если в слове существует блок не менее чем из трёх согласных подряд и эти согласные — не одна и та же буква.

Например, следующие слова набраны с опечатками: «hellno», «hackcerrs» и «backtothefutttture»; следующие слова набраны без опечаток: «helllllooooo», «tobeornottobe» и «oooooo». Редактор Beroffice при обнаружении слова с опечаткой вставляет минимальное количество пробелов в слово (разделяя его на несколько слов) так, что каждое из получившихся слов набрано без опечаток.

Реализуйте эту функциональность редактора Beroffice. Считайте гласными только буквы 'a', 'e', 'i', 'o' и 'u'. Все остальные буквы следует считать согласными.

**Входные данные:** В единственной строке входных данных содержится непустое слово, состоящее из строчных букв английского алфавита.

**Выходные данные:** Выведите заданное слово без изменений, если оно не содержит опечаток. Если слово содержит хотя бы одну опечатку, то вставьте в него наименьшее количество пробелов так, что каждое из получившихся слов не содержит опечаток. Если решений несколько, то выведите любое из них.

**Примеры**

**входные данные**

hellno

**выходные данные**

hell no

**Листинг программы**

**#5**

**text = str(input())**

**def task5(text):**

**print(text)**

**length = len(text)**

**i, count = 0, 0**

**for i in range(length):**

**count+=1**

**if(text[i] in ('a', 'e', 'i', 'o', 'u') ):**

**count=0**

**if (i>=2 and text[i]!=text[i-1] and count == 3):**

**print(" ", end='')**

**count=1**

**print(text[i], end='')**

**i+=1**

**print('', "\n")**

**#тесты**

**test1 = "hellno"**

**test2 = "hackcerrs"**

**test3 = "backtothefutttture"**

**test4 = "helllllooooo"**

**test5 = "bckthftrclmnfxz"**

**test6 = "aeiouaeiou"**

**#вывод**

**# task5(test1)**

**# task5(test2)**

**# task5(test3)**

**# task5(test4)**

**# task5(test5)**

**# task5(test6)**

**task5(text)**

# Задача 6

**Условие задания**

В ряд расположены n жемчужинок. Пронумеруем жемчужинки целыми числами от 1 до n слева направо. Жемчужинка номер i имеет тип ai.

Последовательность подряд идущих жемчужинок будем называть подотрезком. Подотрезок будем называть хорошим, если в нём есть пара жемчужинок одинакового типа.

Вам требуется разбить исходный ряд жемчужинок на наибольшее количество хороших подотрезков. Обратите внимание, что каждая жемчужинка должна попасть ровно в один подотрезок разбиения.

Рекомендуется для ввода и вывода данных использовать функции scanf, printf в языке C++, поскольку они работают значительно быстрее потоков cin, cout. Аналогично, рекомендуется использовать классы BufferedReader, PrintWriter вместо Scanner, System.out в языке Java.

**Входные данные** В первой строке находится целое число n (1 ≤ n ≤ 3·105) — количество жемчужинок в ряду.

Во второй строке находятся n целых чисел ai (1 ≤ ai ≤ 109) — тип i-й жемчужинки.

**Выходные данные** В первой строке выведите целое число k — наибольшее количество подотрезков в разбиении ряда жемчужинок.

В каждой из следующих k строк выведите по два целых числа lj, rj (1 ≤ lj ≤ rj ≤ n) — номера самой левой и самой правой жемчужинки в j-м подотрезке.

Выведенный набор должен образовывать разбиение ряда жемчужинок на хорошие подотрезки, то есть каждая жемчужинка исходного ряда должна попасть ровно в один подотрезок и каждый подотрезок должен содержать пару жемчужинок одинакового типа.

Если существует несколько оптимальных решений вы можете вывести любое из них. Подотрезки можно выводить в любом порядке.

Если исходный ряд невозможно разбить на хорошие подотрезки, выведите одно число "-1".

**Примеры**

**входные данные**

5

1 2 3 4 1

**выходные данные**

1

1 5

**Листинг** **программы**

**#6**

**n = int(input())**

**pearls = list(map(int, input().split()))**

**def task6(n, pearls):**

**a = pearls**

**arr=[]**

**count = 0**

**flag = False**

**if(len(pearls)!=n or n < 2):**

**flag=True**

**for i in range(len(a)-1):**

**if(i+1 not in arr):**

**for j in range(len(a)-1,i,-1):**

**if (a[i]==a[j] and j+1 not in arr and i+1 not in arr):**

**count+=1**

**arr.append(i+1)**

**arr.append(j+1)**

**if (flag):**

**print(-1)**

**else:**

**print(count)**

**for k in range(0,len(arr)-1,+2):**

**print(arr[k], arr[k+1])**

**task6(n, pearls)**

# Задача 7

**Условие задания**

Студентка Мария успешно завершила обучение в престижном вузе и теперь гуляет на выпускном. Студенты много мечтают о красивой жизни, поэтому одногруппники Маши решили соорудить небольшую пирамидку из одинаковых бокалов для шампанского. Получилась пирамида высоты n, такая что на самом верху стоял 1 бокал, на уровне ниже — 2 бокала, ещё ниже — 3 бокала и так далее. В последнем, самом нижнем уровне было ровно n бокалов.

Машин одногруппник Влад вызвался наполнять пирамиду шампанским. Он тысячу раз видел в кино, как эффектно распространяется по бокалам шампанское, перетекая с более высоких этажей на более низкие. Он взял бутылку и начал лить её содержимое в бокал, расположенный на верхнем уровне.

За одну секунду Влад выливает в верхний бокал количество шампанского, равное по объёму в точности одному бокалу. Если бокал полностью наполнен, а в него льётся шампанское, то все излишки поровну распределяются между двумя бокалами, на которых расположен данный. Если переполненный бокал стоит на нижнем уровне, то шампанское просто выливается на стол. В рамках данной задачи будем считать, что шампанское растекается по пирамиде мгновенно. Влада интересует, сколько бокалов будут наполнены до краёв через t секунд, когда бутылка подойдёт к концу.

Входные данные В единственной строке входных данных записаны два целых числа n и t (1 ≤ n ≤ 10, 0 ≤ t ≤ 10 000) — высота пирамиды и количество единиц времени, через которое бутылка закончится.

Выходные данные Выведите единственное число — количество наполненных доверху бокалов в пирамиде через t секунд.

**Примеры  
 входные данные** 3 5  
 **выходные данные**  
 4

**Листинг программы**

**#7**

**n, t = list(map(int, input().split()))**

**def task7(n, t):**

**full = 0**

**k = [[0] \* (i + 1) for i in range(n + 1)]**

**k[0][0] = t**

**for i in range(n):**

**for j in range(i + 1):**

**if k[i][j] >= 1:**

**full += 1**

**k[i + 1][j] += (k[i][j] -1) / 2**

**k[i + 1][j + 1] += (k[i][j] -1) / 2**

**print(full)**

**task7(n,t)**

# Задача 8

**Условие задания**

И где здесь телефонные номера?

Дана строка s из строчных букв английского алфавита и число k. Найдите лексикографически минимальную строку t, имеющую длину k, множество букв которой является подмножеством множества букв s и s лексикографически меньше t.

Гарантируется, что ответ существует.

Обратите внимание, что под множеством букв строки подразумевается множество, а не мультимножество. В частности, множество букв строки abadaba это {a, b, d}.

Строка p считается лексикографически меньше строки q, если p — префикс q, не равный q или существует i такое, что pi < qi и для всех j < i выполнено pj = qj. Например, abc лексикографически меньше abcd , abd лексикографически меньше abec, afa лексикографически не меньше ab и a лексикографически не меньше a.

**Входные данные**

В первой строке через пробел заданы два целых числа n и k (1 ≤ n, k ≤ 100 000) — длина строки s и необходимая длина строки t.

Во второй строке задана строка s, состоящая из n строчных букв английского алфавита.

**Выходные данные**

Выведите строку t, удовлетворяющую условиям, описанным выше.

Гарантируется, что ответ существует.

**Примеры**

**входные данные**

3 3

abc

**выходные данные**

aca

**Листинг программы**

**#8**

**n,k=** **list(map(int, input().split()))**

**s=input()**

**def task8(n,k,s)**

**l=list(set([i for i in s]))**

**l.sort()**

**if(k<=n):**

**i=k-1**

**while(s[i]==l[-1]):**

**i-=1**

**t=s[:i] #:i**

**j=0**

**while(s[i]>=l[j]):**

**j+=1**

**t=t+l[j]**

**for j in range(i+1,k):**

**t=t+l[0]**

**print(t)**

**else:**

**print(s+l[0]\*(k-n))**

**task8(n,k,s)**

# Задача 9

**Условие задания**

Вадим очень любит путешествовать. Недавно он услышал о том, что в его городке появился клуб каякеров, и его очень заинтересовал этот вид путешествия. Поэтому Вадим присоединился к группе каякеров и готов отправиться в сплав.

Сейчас группе нужно распределиться по каякам. В группе 2·n человек (включая Вадима), и у них есть ровно n - 1 двухместных и 2 одноместных каяка. Вес i-го человека равен wi, и вес очень важен в сплаве — если разница весов двух человек, сидящих в двухместном каяке, слишком велика, то каяк становится неустойчивым и может перевернуться. И, конечно же, каякеры хотят распределиться по плавсредствам так, чтобы этого не произошло.

Формально, неустойчивость одноместного каяка всегда равна 0, а неустойчивость двухместного каяка равна модулю разности между весами людей, которые заняли этот каяк. Неустойчивость всей группы — это сумма неустойчивостей всех каяков.

Помогите группе определить минимально возможную суммарную неустойчивость!

**Входные данные** В первой строке задано одно целое число n (2 ≤ n ≤ 50).

Во второй строке заданы 2·n целых чисел w1, w2, ..., w2n, где wi — вес человека i (1 ≤ wi ≤ 1000).

**Выходные данные** Выведите минимально возможную неустойчивость группы.

**Примеры  
 входные данные**  
 2  
 1 2 3 4  
 **выходные данные**  
 1

**Листинг программы**

**#9**

**n = int(input())**

**w = list(map(int, input().split()))**

**w.sort()**

**answer = []**

**def task9(n, w):**

**#del w[len(w)-2:len(w)]**

**for i in range(len(w)):**

**for j in range(i+1, len(w)):**

**imbalance = 0**

**weights = []**

**for k in range(len(w)):**

**if k!=i and k!=j:**

**weights.append(w[k])**

**for l in range(1,n):**

**imbalance += weights[2\*l-1] - weights[2\*l-2]**

**answer.append(imbalance)**

**answer.sort()**

**print(answer[0])**

**task9(n,w)**

# Задача 10

Санта-Клаус решил разобрать свою клавиатуру, чтобы её почистить. После того, как он поставил все клавиши обратно, он с ужасом понял, что что-то не так: некоторые пары клавиш перепутаны между собой! Таким образом, Санта-Клаус подозревает, что каждая клавиша либо стоит на своём месте, либо заняла место другой, а та другая — на месте первой.

Для того, чтобы убедиться в этом, найти ошибку и восстановить верное расположение, Санта-Клаус набрал текст своей любимой скороговорки, смотря только на надписи на клавиатуре.

Вам даны любимая скороговорка Санта-Клауса и строка, которая получилась в результате набора. Определите, какие пары клавиш Санта-Клаус мог перепутать. Каждая клавиша должна принадлежать не более чем одной паре перепутанных клавиш.

**Входные данные** Входные данные состоят из двух строк s и t — любимой скороговорки Санта-Клауса и строки, которая получилась после набора скороговорки. Строки s и t непусты и имеют одинаковую длину, которая не превышает 1000, включительно. Обе строки состоят только из строчных латинских букв.

**Выходные данные** Если предположение Санта-Клауса неверно и клавиатура требует починки и её нельзя починить, поменяв местами буквы в нескольких непересекающихся парах, выведите одно число «-1» (без кавычек).

Иначе в первой строке выведите число k (k ≥ 0) — количество пар букв, которые нужно поменять местами. Затем в следующих k строках выведите по две буквы, разделённые пробелом — буквы, которые необходимо поменять местами на клавиатуре. Все выведенные буквы должны быть различны.

Если ответов несколько, выведите любой. Как пары, так и буквы в парах можно выводить в любом порядке.

Каждая буква должна присутствовать не более чем в одной паре. Санта-Клаус считает, что клавиши расположены корректно, если он может набрать на клавиатуре текст своей любимой скороговорки без ошибок.

**Примеры**  
 **входные данные**  
 helloworld  
 ehoolwlroz  
 **выходные данные**  
 3  
 h e  
 l o  
 d z

**Листинг программы**

**#10**

**s = str(input())**

**t = str(input())**

**def task10(s,t):**

**pairs = []**

**setchars = []**

**flag = False**

**for i in range(len(s)):**

**if(s[i]!=t[i] and (t[i],s[i]) not in pairs and (s[i],t[i]) not in pairs):**

**pairs.append((s[i],t[i]))**

**if(s[i] in setchars or t[i] in setchars):**

**flag = True**

**setchars.append(s[i])**

**setchars.append(t[i])**

**#print(setchars)**

**#print(pairs)**

**for i in range(len(s)):**

**if(s[i]==t[i] and s[i] in setchars):**

**flag = True**

**if(flag):**

**print(-1)**

**else:**

**print(len(pairs))**

**i=0**

**while (i<len(setchars)-1):**

**print(setchars[i], setchars[i+1])**

**i+=2**

**task10(s,t)**

**Вывод**

В ходе выполнения курсовой работы были разработаны программы для решения 7 задач.